UNIVERSIDAD
COMPLUTENSE
MADRID

PROYECTO DE INNOVA-DOCENCIA
Convocatoria 2019/2020

Nº de proyecto: 157

Título del proyecto
DISEÑO Y CREACIÓN DE VIDEOJUEGOS MEDIANTE HERRAMIENTAS DE SOFTWARE LIBRE

Responsable del proyecto
DARÍO LANZA VIDAL

Centro
FACULTAD DE BELLAS ARTES

Proyecto Interdepartamental
Departamento de Escultura y Formación Artística
Departamento de Dibujo y Grabado
Departamento de Diseño e Imagen

Memoria de actividades

DISEÑO Y CREACIÓN DE VIDEOJUEGOS MEDIANTE HERRAMIENTAS DE SOFTWARE LIBRE

ÍNDICE

UNIDADES DIDÁCTICAS

Proyecto de Innova-Docencia nº 157.
Departamento de Escultura y Formación Artística. Departamento de Dibujo y Grabado y Departamento de Diseño e Imagen.
Responsable del Proyecto: Darío Lanza Vidal. dlanza@ucm.es.
Convocatoria 2019/2020

1. INTRODUCCIÓN

Resumen
El objetivo del presente proyecto es aproximar tanto a alumnos como a profesores al empleo de herramientas de software libre y gratuito a través de la creación de un videojuego de temática educativa ambientado en la época de las Cuevas de Altamira.
A la finalidad principal -la aproximación del alumno a una industria en desarrollo empleando para su formación herramientas a su alcance que no requieren desembolso económico- habrá de sumarse una labor educativa, pues el videojuego planteado aprovecha la celebración del 150 aniversario del descubrimiento de las Cuevas de Altamira para ambientar su narración en el período paleolítico y a través de ello informar a los jugadores acerca de la vida en este período de nuestra historia.

Palabrabras clave: Software libre; Videojuegos; Concept art; Arte rupestre; Diseño.

Descripción del proyecto

Según el *"Libro blanco del desarrollo español de videojuegos 2018",* el videojuego es el principal motor del entretenimiento global y representa una industria que ha sido capaz de generar 134.900 millones de dólares en 2018, presentando un crecimiento anual del 10,9%. Una industria con un desarrollo de estas dimensiones supone un horizonte profesional de indudable valor para una generación de estudiantes que pueden hallar en la industria del videojuego una salida profesional de enorme futuro. Sin embargo, vemos con frecuencia que el alumno percibe un abismo insalvable para aproximarse a esta industria al asumir que la preparación para el trabajo en dicho sector exige inevitablemente un gran desembolso económico en hardware y software específicos. Con el presente proyecto pretendemos salvar dicho espacio y aproximar al alumno a esta poderosa industria emergente demostrando la posibilidad de creación de un videojuego completamente funcional mediante exclusivamente softwares libres y gratuitos.
Surgido como continuación de otros dos PIMCDs sobre las posibilidades docentes y creativas del software libre, los miembros participantes en este proyecto, todos ellos expertos en materias de tecnologías digitales aplicadas a los ámbitos del diseño, la animación y el videojuego, realizarán el diseño y creación de un videojuego desde su diseño inicial hasta su exportación a las diferentes plataformas de juego, empleando durante el proceso únicamente softwares libres o gratuitos y produciendo, al mismo tiempo que el desarrollo del videojuego, una memoria de trabajo que servirá a alumnos o profesores como manual con el que aprender tanto la creación de un videojuego como el uso práctico de los softwares empleados. Ambos elementos, el videojuego final y la memoria/manual, se publicarán en abierto para garantizar la mayor difusión de los desarrollos y metodologías investigadas durante el proyecto.
Por otro lado, la temática del videojuego es también un factor esencial para este proyecto, pues creemos que los videojuegos pueden y deben cumplir una función educativa para los jugadores, quienes en muchas ocasiones son niños y adolescentes. Buscamos, por tanto, que además de servir esta experiencia como un poderoso material docente para profesores y futuros profesionales, el propio videojuego en sí cumpla además una función educativa y formadora, más allá de su valor como entretenimiento. Por ello, y aprovechando el 150 aniversario del descubrimiento de las Cuevas de Altamira, hemos escogido que nuestro videojuego utilice la estética de dichas pinturas rupestres y narre la historia de un personaje paleolítico que debe enfrentarse a la vida prehistórica, reforzando ideas como el descubrimiento del fuego, la producción de herramientas, la caza de bisontes… De este modo ayudamos a formar y difundir conocimiento sobre dicho periodo de la historia de la humanidad utilizando para ello un medio que resulta muy atractivo a las nuevas generaciones, acercándoles la cultura en un formato tradicionalmente de entretenimiento.
En línea con esta vocación educativa, el proyecto plantea además la celebración de dos conferencias: una acerca de Arte Rupestre por parte de los restauradores de las Cuevas de Altamira, y otra relacionada con la industria del videojuego a cargo de un/a profesional de este sector, que explicará a los alumnos interesados las cuestiones más importantes para aproximarse a esta industria, tanto a nivel creativo como profesional.
Un proyecto, en resumen, que busca dar difusión a las tecnologías libres como herramientas de gran interés para el creativo actual, que ayuda a aproximar a alumnos a la floreciente industria del videojuego, que proporciona material docente para profesores de asignaturas tecnológicas, que persigue un objetivo educativo al desarrollar una narrativa histórica con fines culturales y que prolonga su vocación educativa en la forma de dos conferencias de gran interés.

Objetivo del proyecto y propuestas de valor del mismo

En base a los programas del Grado de Bellas Artes incluidos en asignaturas de perfil tecnológico, como “Tecnologías Digitales” o “Animación 2D/3D” del Grado en Diseño de Videojuegos, asignaturas que son impartidas por los miembros del presente equipo, este proyecto de innovación educativa propone los siguientes objetivos:

Objetivos Generales:
- Introducir el uso y manejo de herramientas de diseño gráfico, animación, programación, edición de audio y creación de videojuegos, mediante software libre.
- Comprender y utilizar los fundamentos de las tecnologías digitales, estrategias, métodos y procesos de trabajo.
- Adecuar la tecnología disponible a los objetivos en cada proceso de diseño y creación.
- Aprender técnicas digitales por medio de software libre, aplicadas desde el concept art, al diseño y finalmente al desarrollo de videojuegos.
- Capacitar al alumno para adquirir conocimientos sobre la estructura de la industria del videojuego, así como la ubicación y configuración de los centros de toma de decisiones relativas a la misma.
- Capacitar al estudiante para la resolución de problemas de forma autónoma, creativa e innovadora.
- Fomentar la investigación y la experimentación a través de la aplicación de la tecnología digital, con el uso del software libre.
- Aplicar profesionalmente tecnologías específicas.
- Utilizar las herramientas apropiadas para los lenguajes artísticos propios.

Objetivos Específicos:
- Establecer un plan de reuniones mensuales para concretar los objetivos generales, en función de las particularidades técnicas de cada uno de los programas.
- Realizar unidades didácticas de los programas de software libre que vayamos trabajando, en base a un enfoque práctico. Investigaremos las capacidades de programas como Krita o GIMP para el diseño de los fondos es ítems, trabajaremos con Inkscape en el diseño vectorial de los personajes, emplearemos el lenguaje de programación C# para la programación de los scripts y funcionalidades del videojuego y Audacity para la edición del audio, y finalmente utilizaremos Unity para la animación de los personajes y el montaje del videojuego.
- Generar recursos on-line mediante el desarrollo de una página web que contenga las unidades didácticas con las casos prácticos realizados a través de los programas de software libre empleados. Se pretende compartir los resultados no sólo con la comunidad educativa sino también con el conjunto de la sociedad digital, tal y como plantea la filosofía del software libre en su interés por compartir conocimiento.
- Documentar, recopilar, y actualizar toda la información online de los software investigados, incluyendo tutoriales, proyectos y trabajos artísticos, en la forma de una memoria escrita y una página web de libre acceso.
- Realizar una tabla o glosario de herramientas de acceso libre sobre diseño y desarrollo de videojuegos, así como de bibliotecas de recursos libres online.

Justificación e idoneidad de los recursos necesarios

Los miembros que componen el grupo de trabajo están relacionados con las áreas de conocimiento sobre las que este proyecto va a trabajar y sus principales líneas de trabajo se hallan comprometidas con la innovación y la creación tecnológica. En su mayoría han participado en el anterior PIMCD 209 de 2016, "Dibujo y pintura digital. Herramientas de software libre para la creación artística" y presentan el proyecto actual con el propósito de continuar y profundizar en dicha investigación.
Los cinco miembros del equipo pertenecen a la Facultad de Bellas Artes de la UCM. Darío Lanza Vidal, Carmen Pérez González, María de Iracheta Martín y Lara Sánchez Coterón son personal PDI en la Facultad de Bellas Artes, a quienes se sumaron Borja Jaume Pérez, doctorando y becario de colaboración del Departamento de Dibujo y Grabado, Marcos Casero Martín, investigador predoctoral y colaborador docente en el Departamento de Dibujo y Grabado y Yago Cordero Donemech como colaborador en el diseño de sonido.
El responsable del proyecto, Darío Lanza Vidal, es PDI del Departamento de Escultura y Formación Artística, experto en matte painting y en creación artística contemporánea a través de tecnología 3D. Ha participado en el desarrollo de software para el renderizado de imágenes fotorrealistas y trabajado como profesional reconocido en rendering en la empresa Next Limit, desarrollando software de efectos especiales para la industria cinematográfica.
Todos los profesores cuentan con experiencia en la docencia tecnológica. Carmen Pérez González es Directora del Departamento del Dibujo I e imparte la asignatura de dibujo animado “Stop-motion Del carbón al pixel”. María De Iracheta es profesora de la asignatura “Tecnologías Digitales” e impartió clase de concept art y animación en la Escuela Universitaria ESNE, adscrita a la Universidad Camilo José Cela. Darío Lanza también imparte “Tecnologías Digitales”, la asignatura “Materiales y tecnologías en la escultura” del Master en Investigación en Arte y Creación y comparte, junto con Carmen Pérez González, la asignatura “Animación 2D/3D” del

Grado en Desarrollo de Videojuegos. Lara Sánchez Coterón imparte “Espacio Virtual” y “Diseño en los nuevos medios” en el Grado de Diseño de la Universidad Complutense y ha desarrollado una Tesis Doctoral sobre la creación de videojuegos, titulada “Arte y videojuegos: Mecánicas, estéticas y diseño de juegos en prácticas de creación contemporánea”.
Todos los profesores de la Facultad de Bellas Artes que participan en este proyecto de Innovación Docente cuentan con evaluaciones positivas por parte del Programa Docentia. Tanto Darío Lanza, responsable del grupo, como Lara Sánchez Coterón cuentan una evaluación docente Muy Positiva en el pasado curso 2017/2018, María de Iracheta ha recibido cuatro evaluaciones Excelentes y dos Muy Positivas, y Carmen Pérez González ha obtenido una evaluación Excelente correspondiente al período 2014/2017 y el Premio UCM a la Excelencia Docente. Estas circunstancias hacen a este equipo idóneo para el desarrollo del presente proyecto, tanto por su participación en anteriores PIMCDs afines, su trayectoria docente en materias de creación tecnológica, su experiencia profesional e investigadora en las áreas del diseño, la tecnología, la animación y los videojuegos, así como por su apoyo a las herramientas libres y recursos en abierto como vía para la universalización de la formación.

Impacto esperado e indicadores propuestos para medirlo en relación con los objetivos formulados

Este proyecto de innovación docente pretende potenciar la calidad y diversidad de la información que se ofrece al alumno.
Al trabajar con software libre, el coste tecnológico es prácticamente cero. Es de dominio público y su descarga se inserta dentro de la legalidad. Permite al usuario utilizarlo libremente, mejorarlo, modificarlo y adaptarlo a sus necesidades. Presenta la ventaja de no requerir el pago de una licencia por cada computadora en que se instale, lo que permite al centro educativo reducir el coste del material tecnológico en las aulas al tiempo que a los estudiantes descubrir y utilizar herramientas de última generación en casa a coste cero.

Viabilidad

La viabilidad del proyecto queda justificada en el plan de actuación del grupo a través del cronograma propuesto, con el propósito de cumplir las actividades enunciadas, que incluyen desarrollo de unidades docentes, conferencias sobre el tema objeto de investigación y talleres de formación tecnológica.

Transferencia

El presente proyecto busca incrementar la difusión del conocimiento en torno al software libre y en concreto sobre las herramientas de diseño y creación de videojuegos, una disciplina de gran potencial económico que constituye una salida profesional de gran proyección e interés para los alumnos de carreras creativas.
Una vez explorado el potencial de dichas herramientas libres en el citado contexto del videojuego, el equipo investigador redactará una memoria que será publicada a modo de manual cuya vocación es convertirse en un interesante material de estudio tanto para alumnos como para profesores de asignaturas tecnológicas interesados en impartir este tipo de docencia.
Así, la transferencia de conocimiento de este proyecto queda garantizada con la publicación de la citada memoria como recurso libre, maximizando así la difusión de los hallazgos obtenidos por el equipo investigador.
Además, se pretende establecer colaboración con otros centros de innovación docente centrados en el arte y la tecnología, tanto nacionales como internacionales, incrementando aún más la visibilidad de nuestro trabajo.

Recursos humanos. Integrantes del proyecto

- Darío Lanza Vidal: dlanza@ucm.es
- Carmen Pérez González: perezgonzalez@art.ucm.es
- María de Iracheta Martín: mariadeiracheta@pdi.ucm.es
- Borja Jaume Pérez: borjaume@ucm.es
- Lara Sánchez Coterón: larasanc@ucm.es
- Marcos Casero Martín: marcoscasero@ucm.es

UNIDADES DIDÁCTICAS

2. ANIMACIÓN DE PERSONAJES

Carmen Pérez González
Identificador ORCID: 0000-0001-5096-9395

Un dibujo de concepto puede ser una obra de arte y si hablamos de dibujo animado se puede incorporar a su definición la de pertenecer al séptimo arte como parte de su terminología.

El boceto es la representación gráfico-plástica de la idea. En una producción cinematográfica el boceto se manifiesta como un elemento indispensable en la descripción y en la articulación de lo que luego serán los modelos o personajes de todo producto animado.

El boceto será no solo visto como una actividad artística o como un ejercicio gráfico para liberar la tensión creativa, si no también se presenta como poseedor de una funcionalidad y de una carga proyectual evidente y obligada en la creación de los dibujos de los personajes, de sus actitudes y de sus ambientes en la producción de animación o de un videojuego.

Para un creativo el dibujo es la base de toda producción y debe expresar aquello que quiere transmitir (figura 2.1). Ejemplo de una Hoja de Modelo de Personaje para un videojuego realizado con software libre titulado *Altamira Game*.

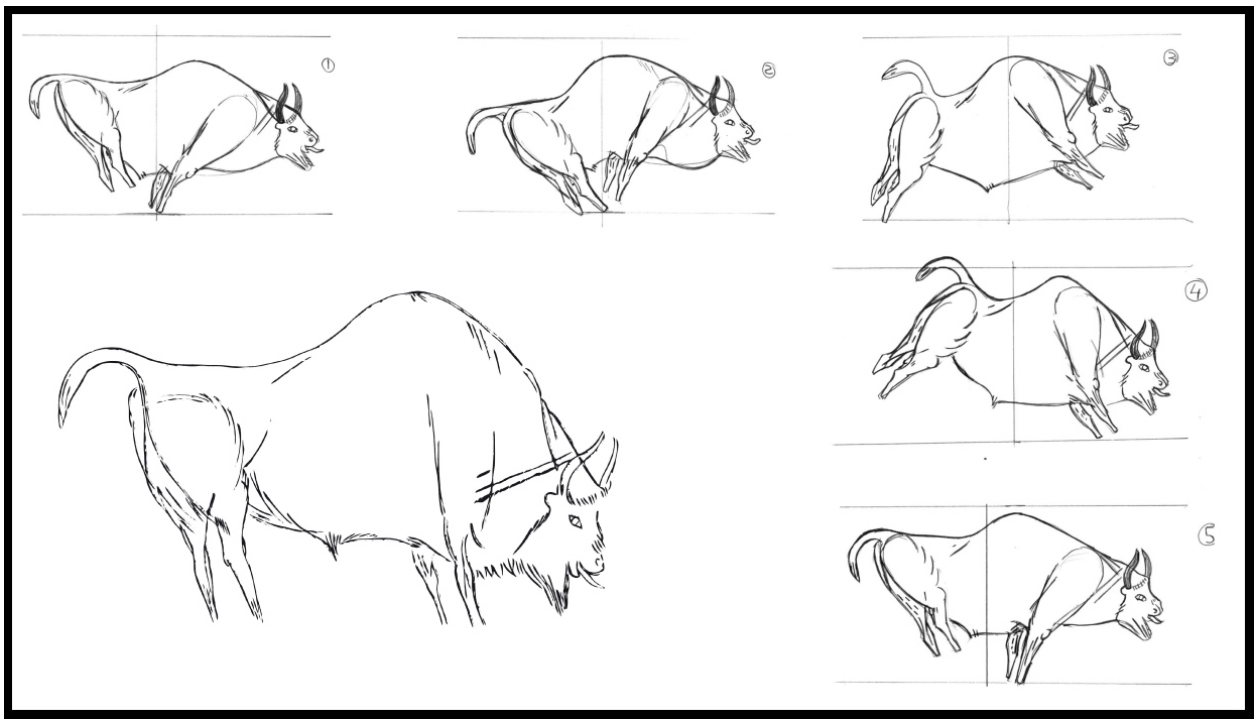


Fig. 2.1. Hoja de Modelo de Personaje para Altamira Game. Bisonte (2019). Fuente: propia

Los inicios al dibujo en movimiento

Si nos adentramos a través de la historia humana, podemos ver que nuestros antepasados comenzaron a representar fenómenos dinámicos y en movimiento, manifestándose tan pronto como adquirieron la capacidad de dibujar.

Las pinturas de la cueva de Altamira (figura 2.2) revelan que nuestros antepasados utilizaron estas pinturas rupestres, no sólo para representar el animal sino también sus movimientos y la narración representada de su caza. El grafismo de los animales se mueve a través de las paredes de la cueva, tratándose de historias gráficas, ilusiones ópticas de movimiento, percibidas cuando la luz de la lámpara reverberaba.

Fig. 2.2. Interior de la “Sala de Polícromos” de la Cueva de Altamira

En las pinturas prehistóricas se puede percibir el fundamento de persistencia retiniana al aplicar el movimiento en la superposición de dos o más imágenes sucesivas de animales en diferentes posturas para crear la ilusión de carrera o de andares como los de la Cueva Chauvet (figura 2.3).

En este trabajo se utiliza una versión contemporánea de este mismo proceso, conocido como técnica de superposición o de papel cebolla, en la que percibe el movimiento pasando muy rápidamente una página delante de otra, utilizando el método llamado de flipeo o flipear.

Fig. 2.3. Cueva Chauvet, Se superpusieron dibujos de dos posturas de un bisonte para representar el movimiento de correr

Los autores Azéma y Rivère, en su libro "Animación en el arte paleolítico: un pre-eco del cine"[1], sugirieron que los europeos de la Edad de Piedra inventaron el taumatropo, uno de los primeros juguetes de animación óptica.

Estos arqueólogos descubrieron que, en varios yacimientos de Francia y España, aparecieron discos de piedra y de hueso en los que se muestran imágenes de un ciclo de salto y carrera de un animal. A estos discos grabados se les metía un cordel por el agujero central que permite girarlo y percibir su movimiento al verlo rápidamente. Cuando la cuerda se hace girar rápidamente, las dos imágenes se unen en una sola y, debido a la ilusión de persistencia retiniana, el cerebro rellena los espacios entre los dibujos, percibiendo movimiento.

Rivère recreo faximiles a partir de estos discos y ambos autores consideran a estos taumatropos paleolíticos como los primeros intentos ópticos de representación del movimiento que nos llevarían a recrear los juguetes ópticos del siglo XIX y a la invención de la cámara cinematográfica (figura 2.4).

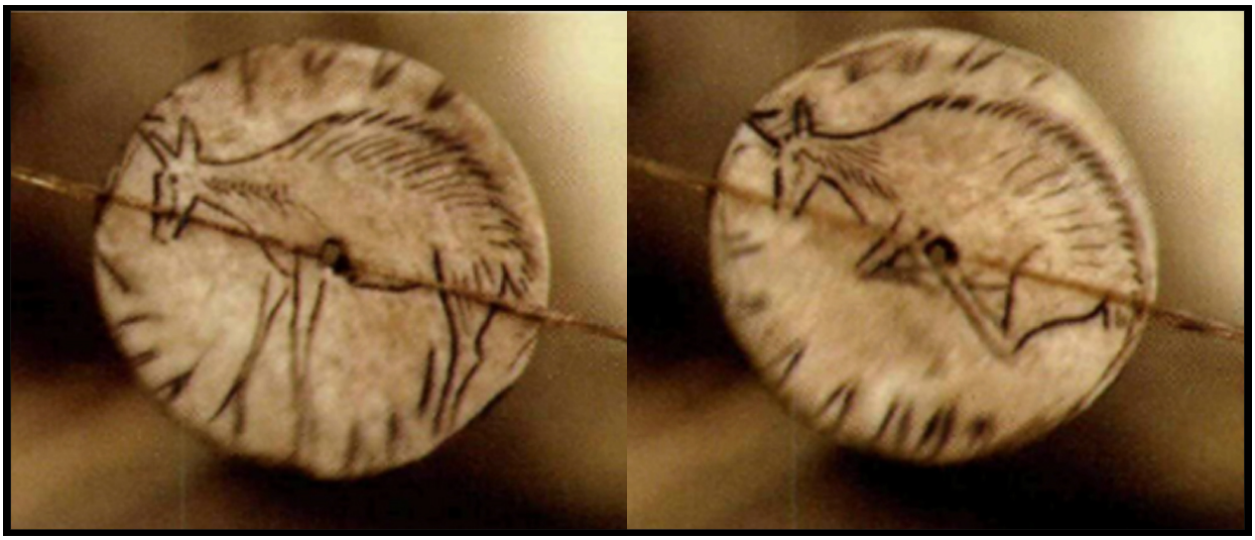


Fig. 2.4. Facsímil de un disco giratorio de Laugerie-Basse realizado por el arqueólogo Rivère que muestra una gamuza en diferentes posiciones claves, o key frames, de un movimiento y que, cuando se hace girar el juguete muy rápidamente, el animal cobra vida

Además de la superposición de imágenes, otro método utilizado por nuestros antepasados fue la representación continuada del movimiento. Este proceso es realizado tomando las diferentes posiciones del animal en el movimiento (figura 2.5). Está técnica ha evolucionado en la animación contemporánea y se conoce como representación de model sheet u hojas de modelo, como son las de los ciclos de andares (figura 2.6), las de los ciclos de carrera y las del ciclo de salto (figura 2.7) entre otras muchas.

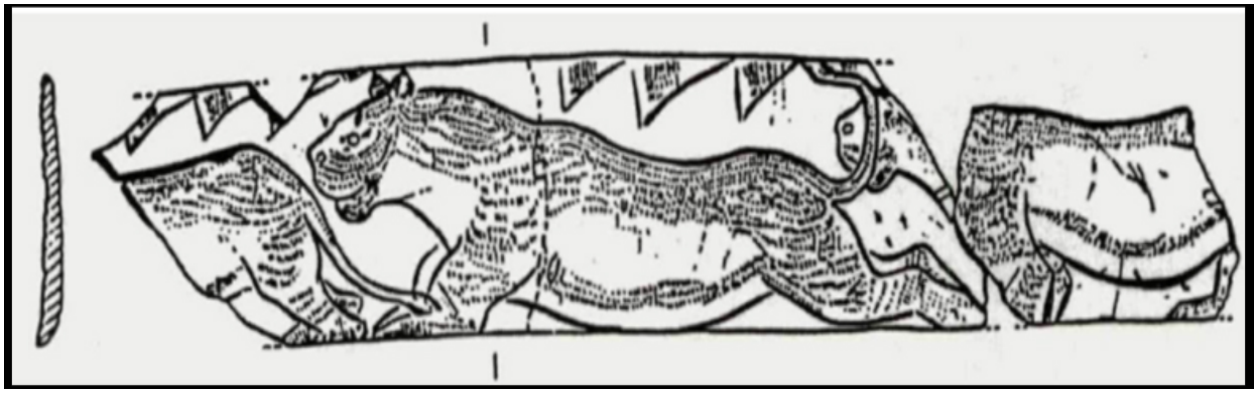


Fig. 2.5. Un dibujo congelado que representa un león corriendo de izquierda a derecha, descubierto en La Cueva Vache

[1] Marc Azéma and Florent Rivère. *Animation in Palaeolithic art: a pre-echo of cinema*. In: Antiquity 86.332 (2012), pp. 316–324.

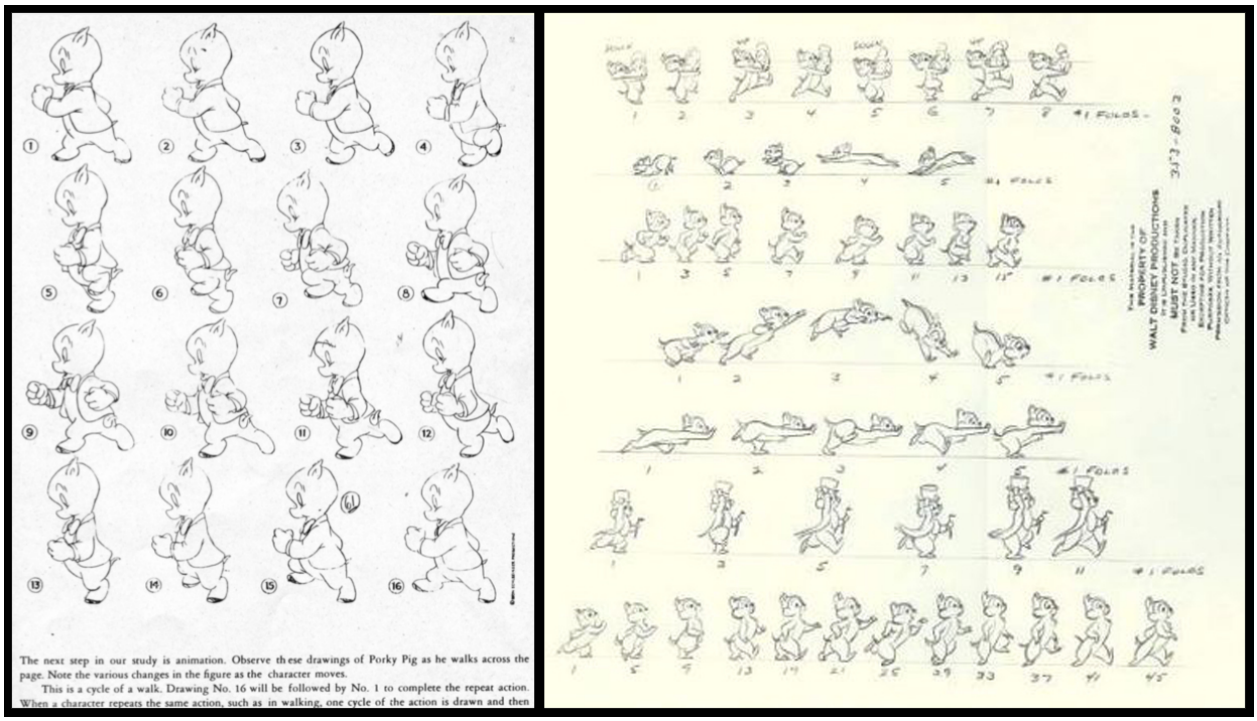


Fig. 2.6. Ciclo de andares de Porky del libro de Alan Dale Bogorad, bajo el título de *Jr's Fun To Draw* (1943, p. 77) y Ciclos de andares, carrera y salto de Chip y Chop Walt Disney

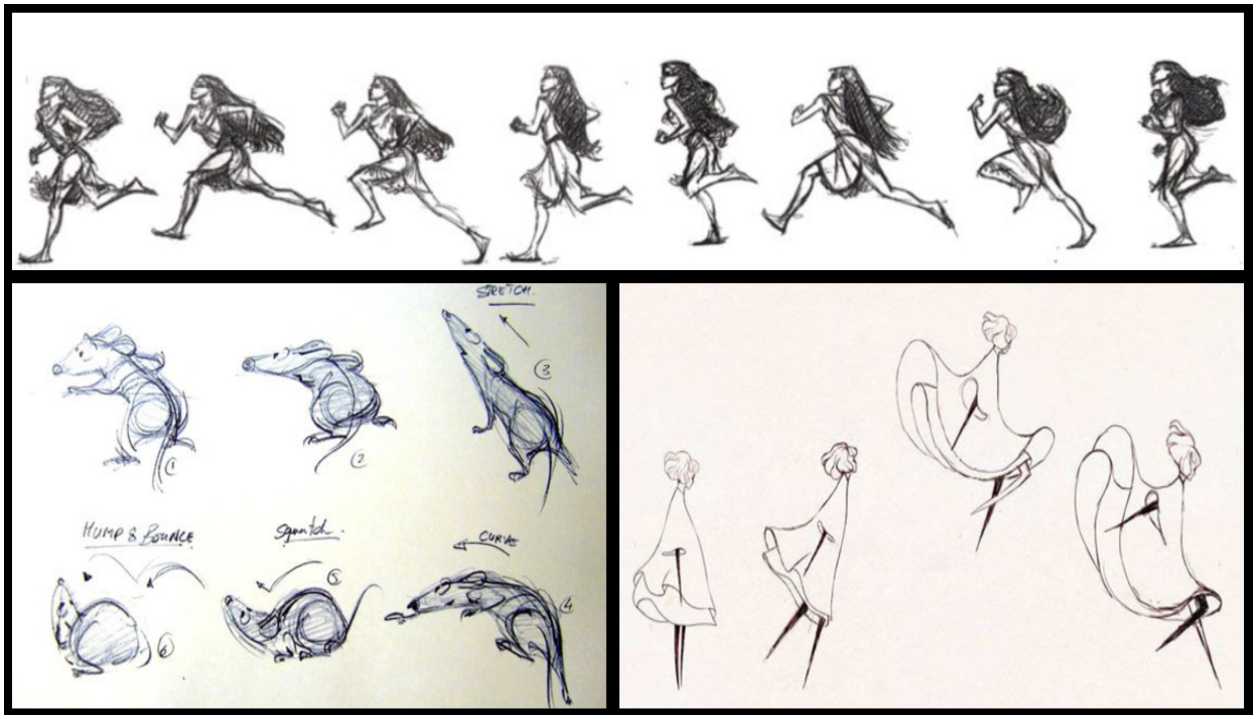


Fig. 2.7. Hojas de ciclos técnicos de movimiento de personajes. Montaje fuente propia a partir de bocetos de Borja Montoro en el ciclo de carrera para *Pocahontas*, Jason Deamer en el ciclos de salto para *Ratatouille* y Ariadna Cervelló en el ciclos de salto para el videojuego *Gris*. Fuente: propia, montaje (2019)

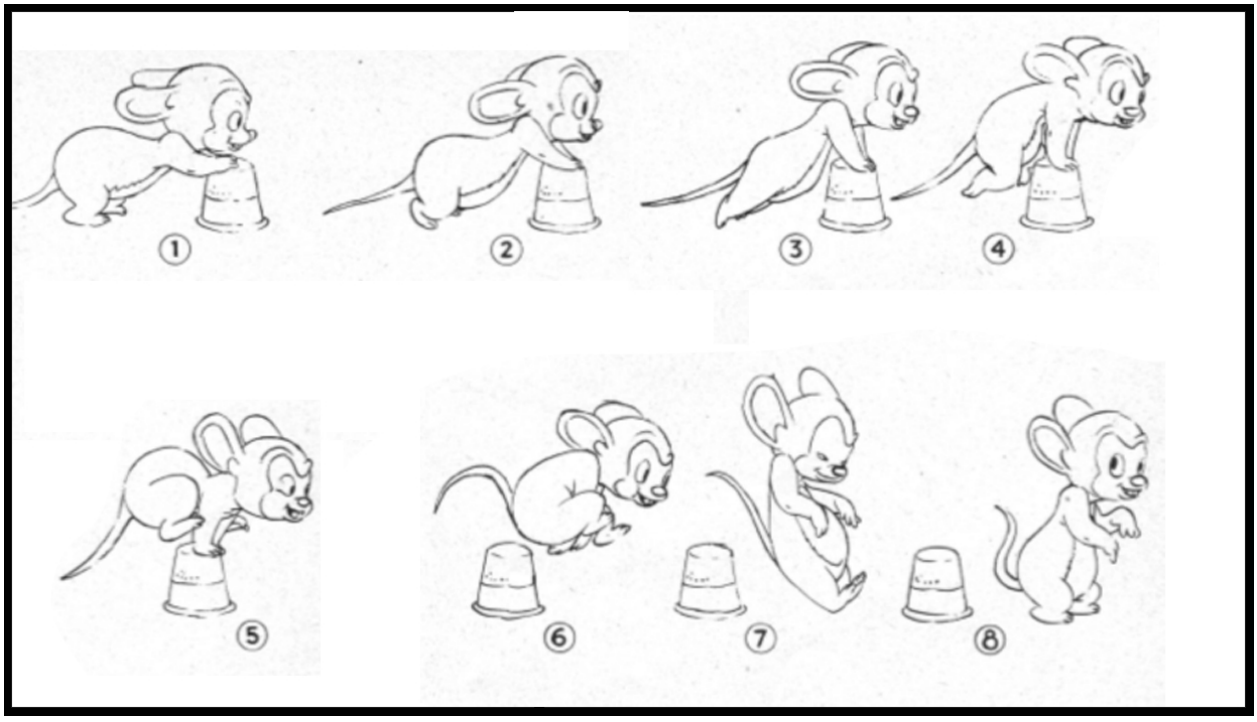


Fig. 2.8. Secuencia de animación de un salto dibujado por la directora de arte Connie Rasinski para *Terrytoons*, incluido en el libro de Alan Dale Bogorad, *Jr's Fun To Draw* (1943, p. 81)

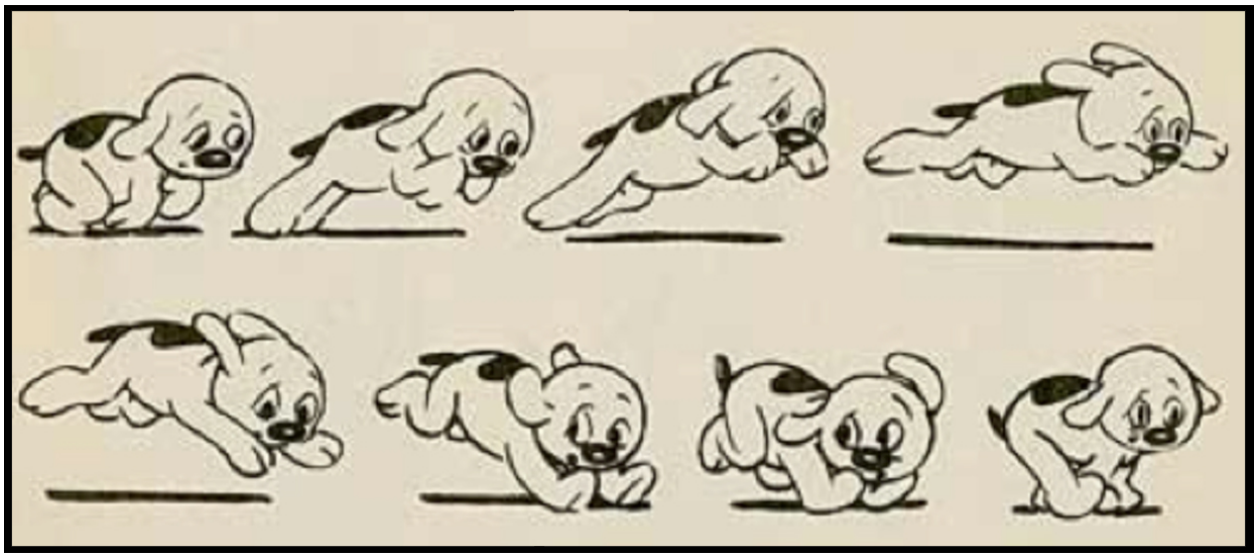


Fig. 2.9. Ciclo de salto de un animal realizado por el director de arte Conrad Rasinski para *Terry Toons* y que aparece en el libro *How to Make animated cartoons nat falk* de Paul Terry (1940)

Desarrollo visual inicial del Arte de Concepto

El objetivo de estas model sheet es hacer factible y viable un posible proyecto gráfico, como puede ser un cómic, una animación, un videojuego o una novela gráfica. Se estaría hablando de un flujo de trabajo realizado en la fase inicial, de la preproducción en entornos de trabajo grupales, y se encuentran bajo la denominación de diseño de concepto o Concept Art. Dentro del espectro del diseño de concepto se encuentran una gran variedad de especialidades, como el diseño de personaje o Character Design, que son los dibujos de los diferentes personajes del mundo a recrear con hojas-tipo.

Como toda fase creativa el dibujante deberá investigar, recopilar y preparar la información que le va a ser necesaria para realizar sus dibujos, sus bocetos y la parte del concepto inicial del producto.

La parte inicial de la investigación será encontrar localizaciones, iluminación, personajes, vestuarios, los movimientos que refuerzan la psicología del personaje, la acción que se va a realizar para después poder recrear las hojas de modelo de los personajes.
En este caso se realizó una búsqueda de imágenes de las pinturas de la Cueva de Altamira (figura 2.2, 2.10, 2.11, 2.12 y 2.13) y también de videojuegos que puedan servir de referencia como: *Guacamelee*[2], *Apotheon*[3]. *Limbo*[4] y *Cuphead*[5].

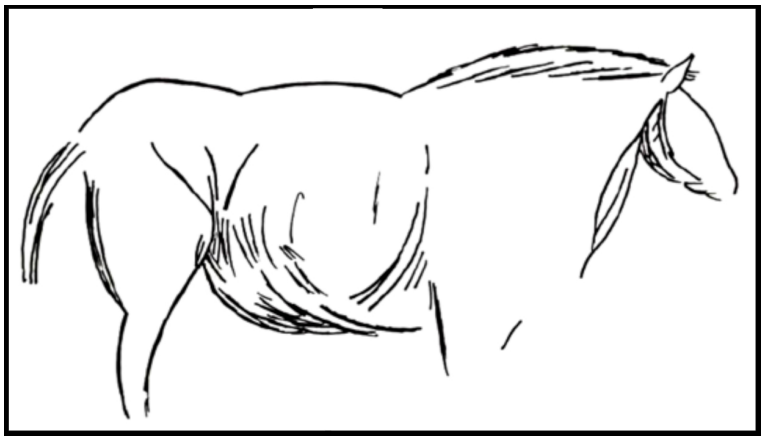


Fig. 2.10. Hola de modelo de personaje para Altamira Game. Caballo percherón

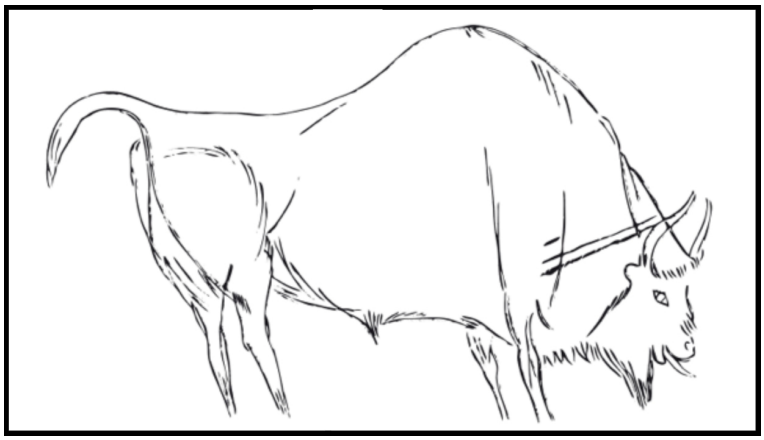


Fig. 2.11. Hoja de modelo de personaje para Altamira Game. Bisonte

2 Quijano, A. (2013, 22 febrero). Guacamelee Trailer. Recuperado 2 enero, 2020, de https://youtu.be/6hGPyHacsG0
3 McGibney, J. (2014, 30 abril). Apotheon. Recuperado 2 enero, 2020, de https://youtu.be/rOvwpj0pw9A
4 Laughton, L. (2010, 10 mayo). LIMBO - Trailer. Recuperado 2 enero, 2020, de https://youtu.be/Y4HSyVXKYz8
5 Moldenhauer, M. (2013, 22 febrero). CupheadTrailer. Recuperado 2 enero, 2020, de https://youtu.be/VUTEdvlUNg0

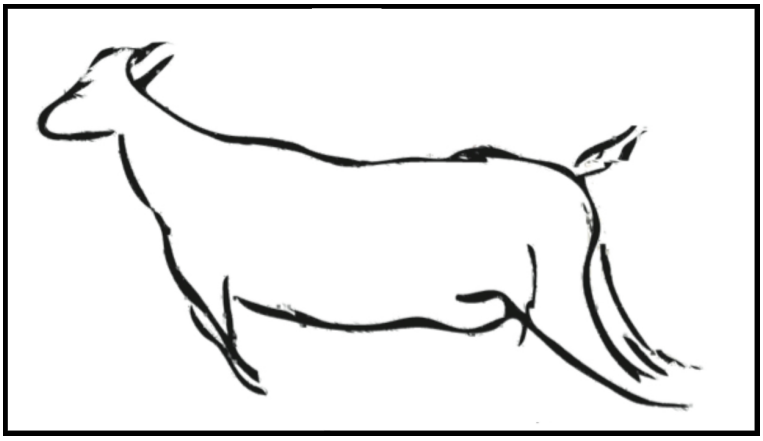


Fig. 2.12. Hoja de modelo de personaje para Altamira Game. Cierva

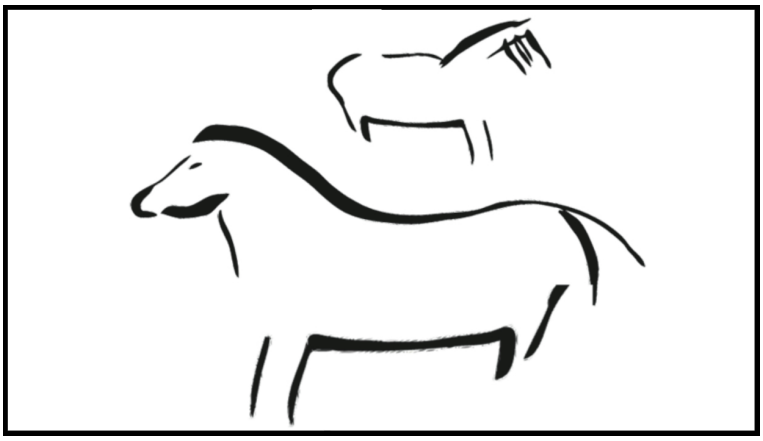


Fig. 2.13. Hoja de modelo de personaje para Altamira Game. Caballo joven

Para realizar los bocetos de los personajes primero se deberá utilizar formas sencillas para su creación, después sus líneas de acción serán evidentes definidoras de su personalidad, además de, plantear una silueta reconocible. El dibujante realizará los fondos de los espacios en los que se va a desarrollar la producción y creará los personajes y los objetos siguiendo una normativa o método unificado entre los animadores empezando todos ellos con sus bocetos.

- Hoja de personaje. Describe con texto el personaje, en algunas ocasiones se incorporan bocetos.
- Hoja de concepto. Plano general del personaje con ampliaciones de características detalladas.
- Hoja de modelo de vestuario, o *model sheets costume*, muestra la indumentaria con la que el personaje vestirá en la producción.
- Hojas modelos de rotación, o *model sheets de turnaround*, muestran los bocetos del personaje en poses de giro. En ellas se dibuja el personaje en cinco poses si es simétrico: de perfil, tres cuartos, frente, espalda y tres cuartos de espalda, en siete poses si es asimétrico y en tres poses si es para un proyecto en 3D (figura 2.14).

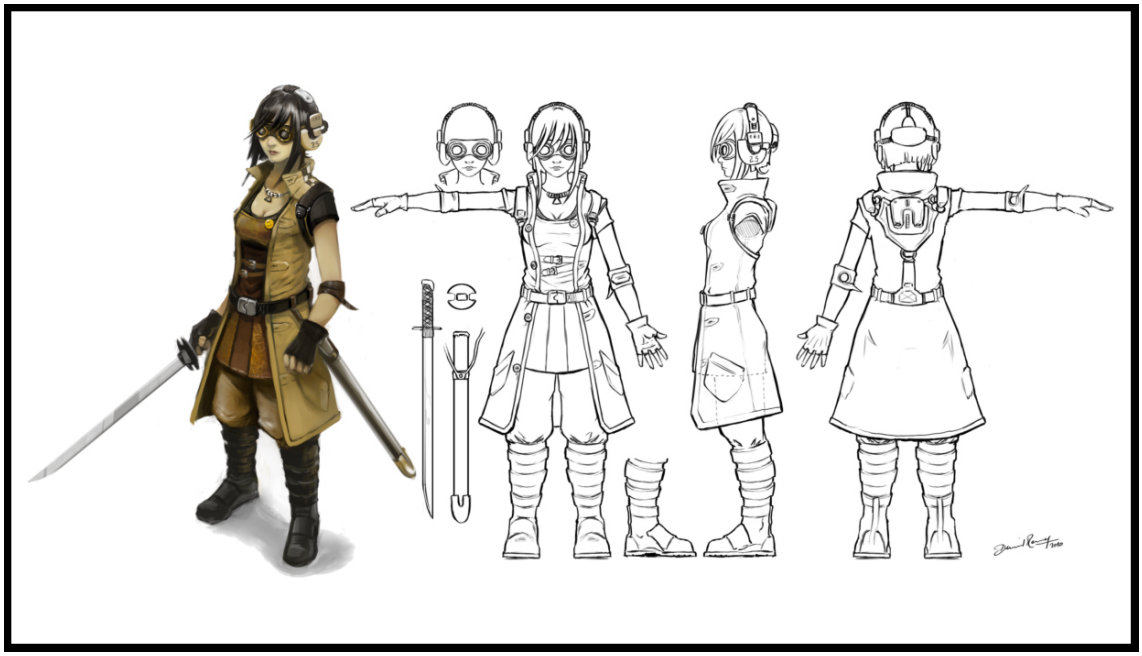


Fig. 2.14. A sample model sheet from the DVD tutorial Chaos&Evolutions. Dibujado por David Revoy (2010)

- Hoja de acción. Presenta la capacidad de movimiento del personaje (figura 2.15).

Fig. 2.15. Hoja de Acción y Expresión para *Cincerella* de Walt Disney, con los dibujantes John Lounsbery, Eric Larson, Norman Ferguson, Les Clark y Marc Davis dibujando a la modelo Helene Stanley para la película (1950)

- En la Hoja de expresiones aparecerán los cuatro colores primarios expresivos: la felicidad, la tristeza, el enfado y el miedo, pero habría que añadir también la amenaza, el desprecio, el dolor, el llanto y la risa (figura 2.16).

Fig. 2.16. Hoja de expresión. Fuente: propia a partir de la hoja de expresión del dibujante Alfie Vann sobre Julio para la película de animación *La ruta hacia El Dorado* (2000)

- Existen también Hojas de Cabezas, de Extremidades (figura 2.17).

Fig. 2.17. Diferentes estéticas de hojas de modelo de extremidades Walt Disney, Mickey Mouse por el dibujante Les Clark, las manos de Hades por Nik Riaanieri, manos de Mushu de Harald Siepermann, las extremidades de Tarzan por Glen Keane. Fuente: propia, montaje (2019)

- Las Hojas de Bocas siguen el convenio que responde a una sincronización entre la locución y el movimiento de los labios, también llamado Lipsync, que ha evolucionado desde el dibujo de cada uno de los fonemas en la línea de tiempo a la automatización de la imagen con el sonido utilizando programas como Toon Boom.

- Estudio de modelo comparativa entre los diferentes personajes de la producción a realizar. Comparativos de los cánones de los diferentes personajes, de las diferentes actitudes, comparando jerarquías y relaciones familiares (figura 2.18).

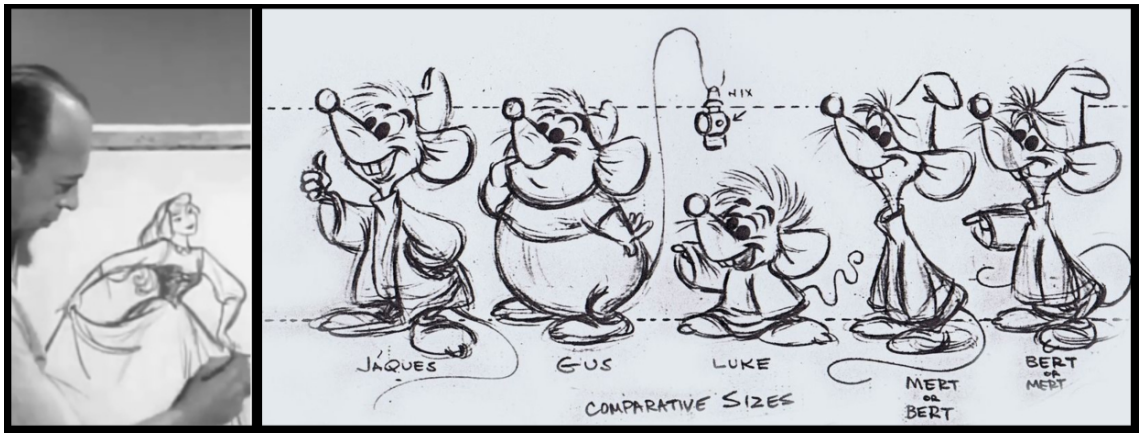


Fig. 2.18. Comparativa de Ratones y Marc Davis dibujando a Helene Stanley para la película de animación Cincerella de Walt Disney (1950). Estudio de modelo de objetos o accesorios

- Hojas modelos de construcción geométrica describe al personaje por medio de medidas y por medio de la geometría.
- Model Sheet de Objetos están los diferentes Accesorios, los Prop u Objetos que utilizan los protagonistas y los Estudios de Decorados.
- Estudio de color que establece su paleta de color característica.
- Hojas modelos de ciclos de movimientos como andares, carreras, saltos o ejercicios atléticos (figura 2.7).
- El Estudio de Color muestra una paleta de color característica para cada uno de los protagonistas (figura 2.19).

Fig. 2.19. Estudio a Color a partir de las Hojas de Comparativa de Chris Sassaki para Monstruos S.A. (2001). Fuente: propia, montaje (2019)

- Las Hojas de Color de fondos detallan una la paleta para la luz, para las medias tintas y para las sombras (figura 2.20).

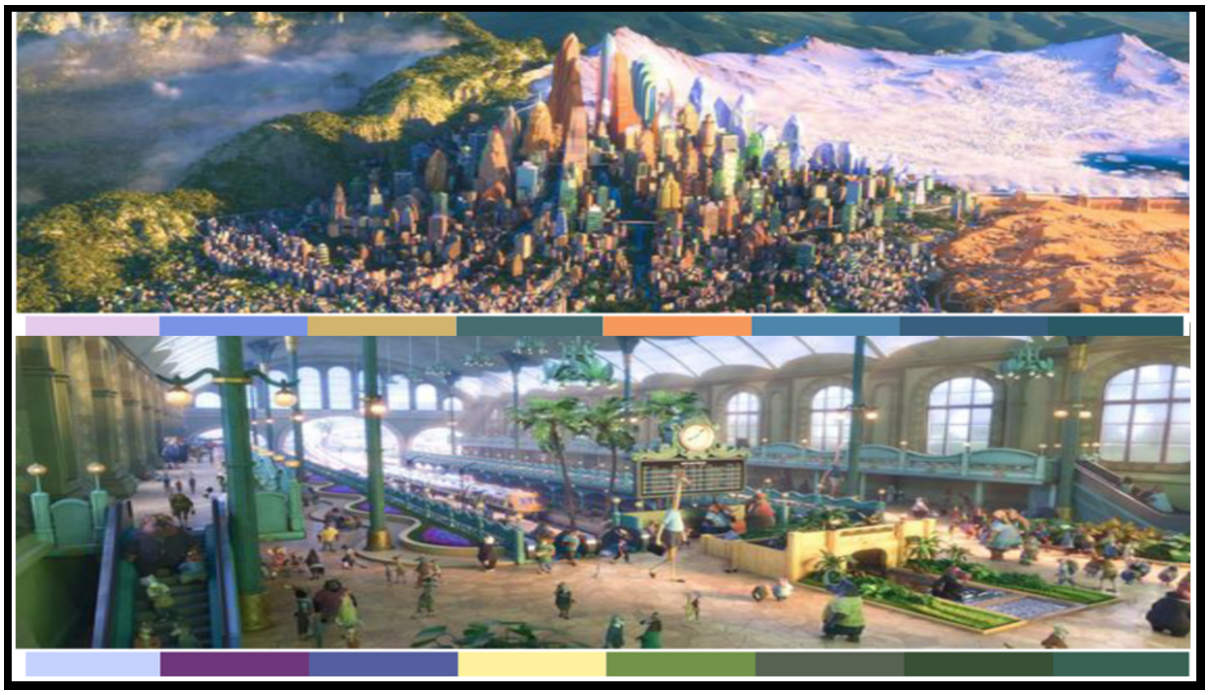


Fig. 2.20. Estudio de Color de Hojas de Fondo a partir de los fondos del artista Cory Loftis para la película de animación Zootropolis (2016). Fuente: propia, montaje (2019)

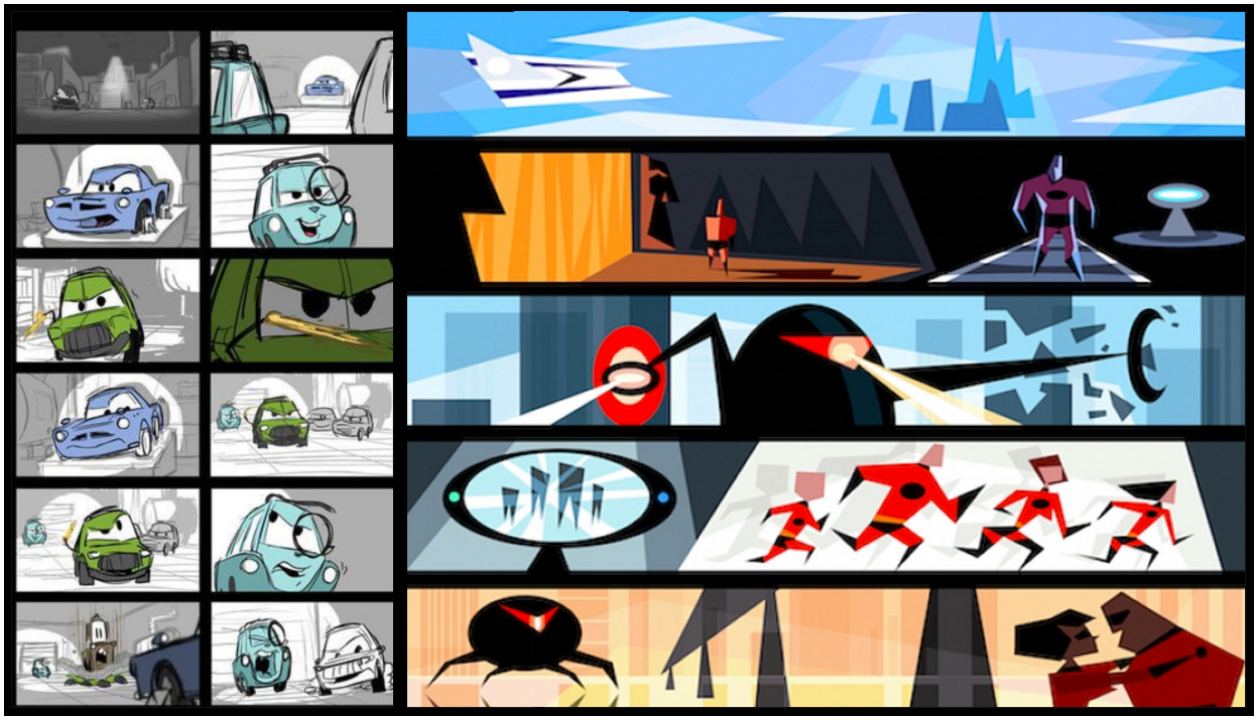


Fig. 2.21. Estudio de Colour Script Matthew Luhn como storyborcista en las producciones de Cars (2006) y Max Brace como storyborcista en las producciones de los Increíbles (2004). Fuente: propia, montaje (2019)

- Por último, a la hora de contar la historia en imágenes se especifican las Hojas de Guía de Color también llamado Colour Script (figura 2.21).

- Cada dibujante es un caso particular y elegirá contar la historia con dibujos señalando los encuadres y las angulaciones precisas, con un formato de Storyboard, Sketchbook o con un formato de Cómic. En el proyecto que nos ocupa se han necesitado, además de las fundamentales hojas de personaje (figuras 2.10 a 2.13), las hojas de concepto de movimiento (figuras 2.22 a 2.25) y los dibujos individuales de los ciclos utilizando la técnica de flipeo para percibir el movimiento con la mesa de luz y con sus registro de animación (*peg-bars*) (figuras 2.26).

En el momento de la pre-producción se analizan sobre todo el dibujo y la concepción de los elementos del producto. Éstos son revisados y entregados en momentos puntuales, en las reuniones del trabajo grupal, donde se tendrán en cuenta los comentarios o las opiniones de los diferentes miembros del grupo.

Programas gratuitos para vectores

El software vectorial posee suficiente versatilidad como para crear ilustraciones fácilmente, pero la gran ventaja es que se puede realizar gráficos que permiten su escalado sin pérdida de calidad. Existe software vectorial gratuito por internet. Uno de ellos es Gravit Designer. Otros programas gratuitos que se puede usar online son Autotracer.org y Svg Edit. Un programa online para principiantes que necesiten pasar imágenes a vector es Vectr.

Autotracer.org será el utilizado en el presente proyecto y permite crear y editar imágenes y convertirlas en vector gratuitamente (figura 2.27). Si simplemente necesitas pasar de un dibujo a vector estos cuatro programas son válidos para ello.

Al trabajar con un programa que genera gráficos vectoriales creas dibujos no como píxeles sino como geometría, como ecuaciones. Comúnmente se presentan bajo la extensión SVG, acrónimo de *scalable vector graphics*. Son gráficos pensados para su uso en Internet ya que hacer fórmulas matemáticas y no mapa de bits, por tanto, son rápidos y potentes a la vez que estéticos, y pueden ser escalados sin perder resolución. En este proyecto se han vectorizado los dibujos para poder con ellos pasar a la fase de color.

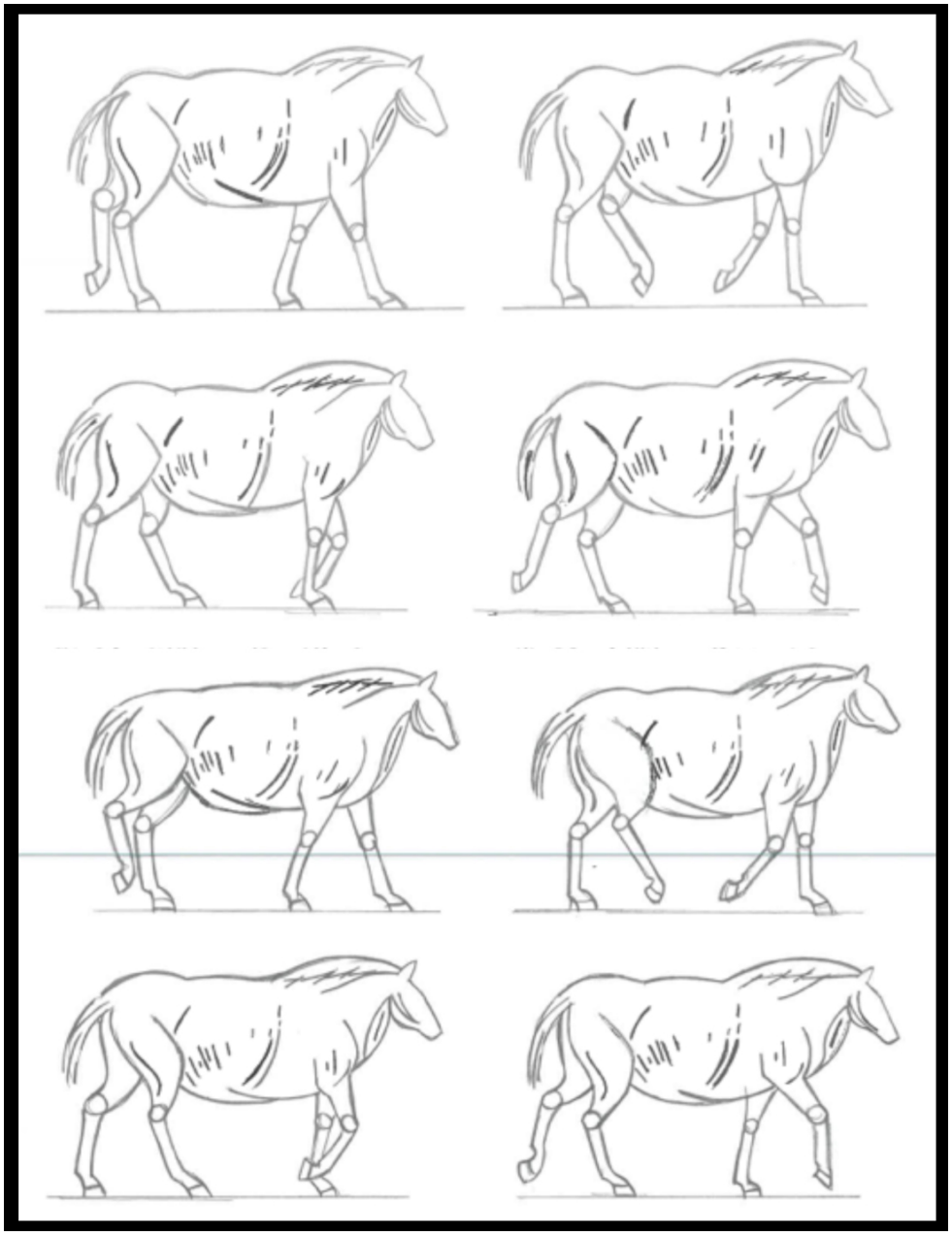


Fig. 2.22. Hoja de modelo de andares para Altamira Game. Caballo percherón

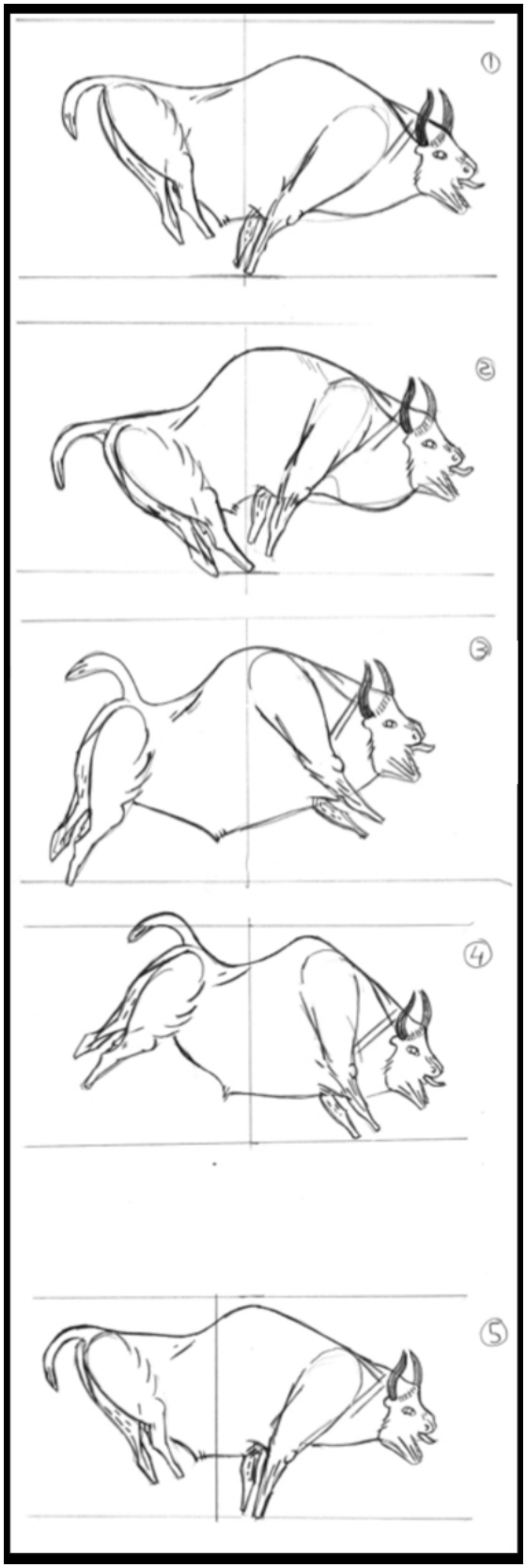


Fig. 2.23. Hoja de modelo de salto para Altamira Game. Bisonte

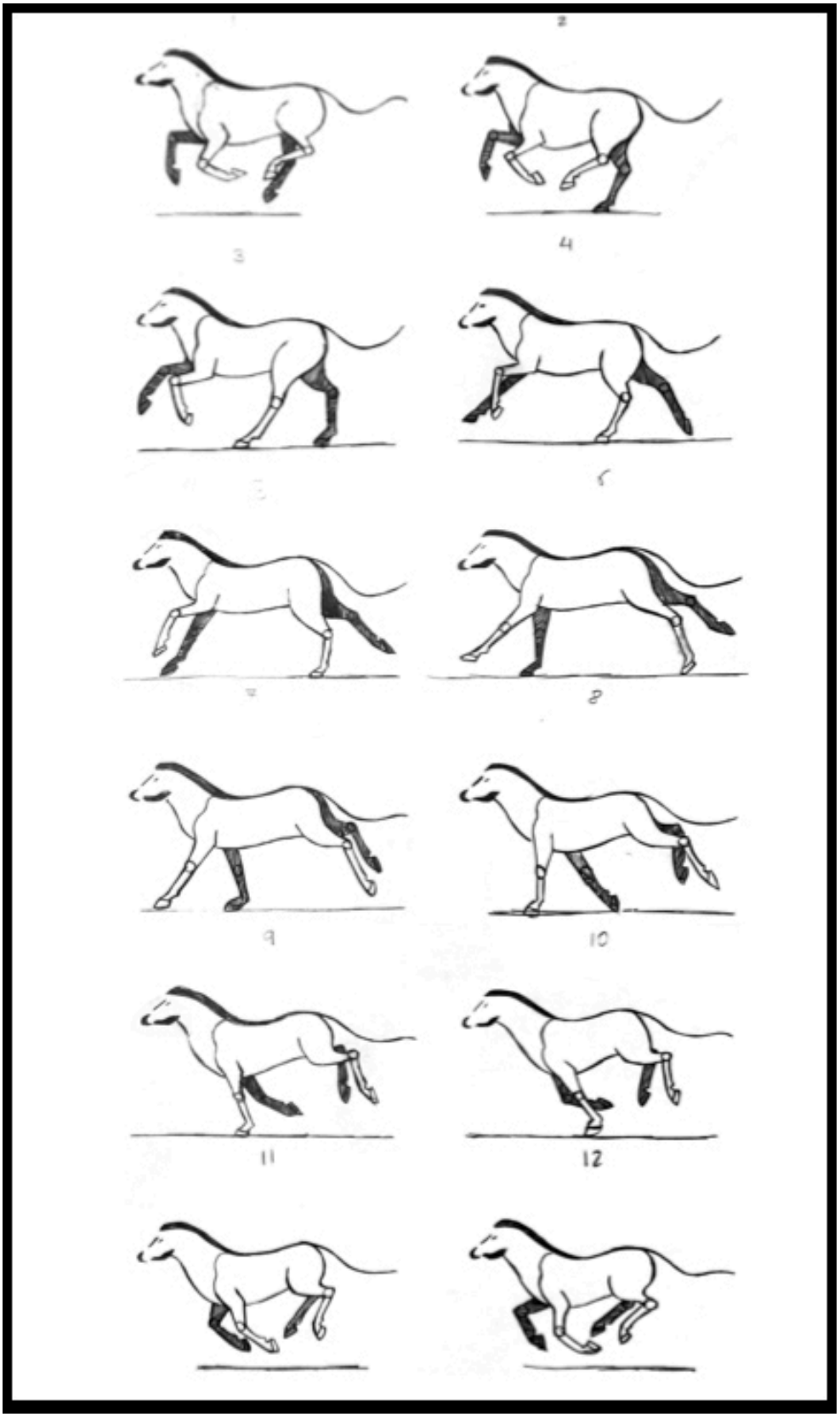


Fig. 2.24. Hoja de modelo de galope para Altamira Game. Caballo

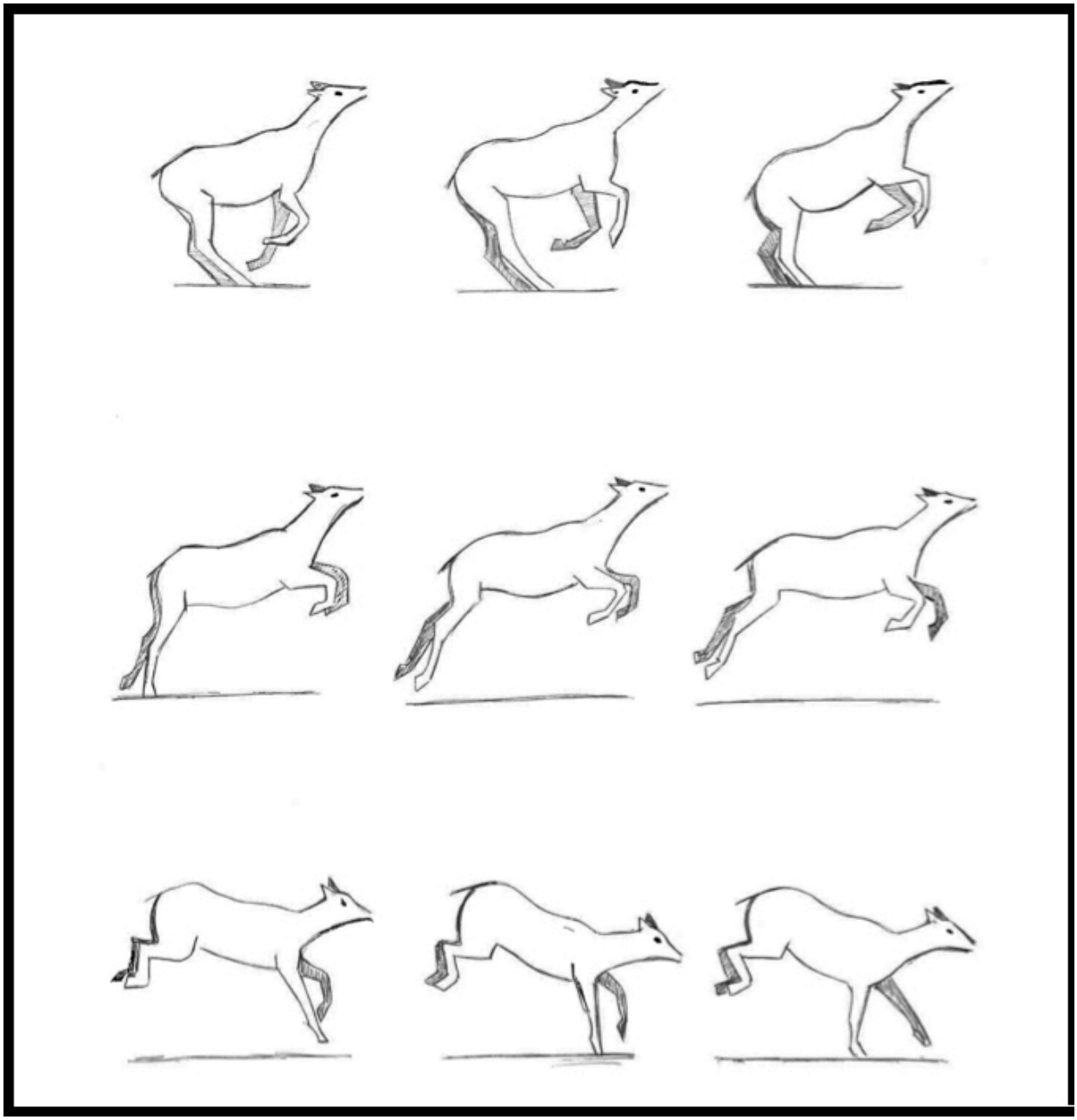


Fig. 2.25. Hoja de modelo de galope para Altamira Game. Cierva

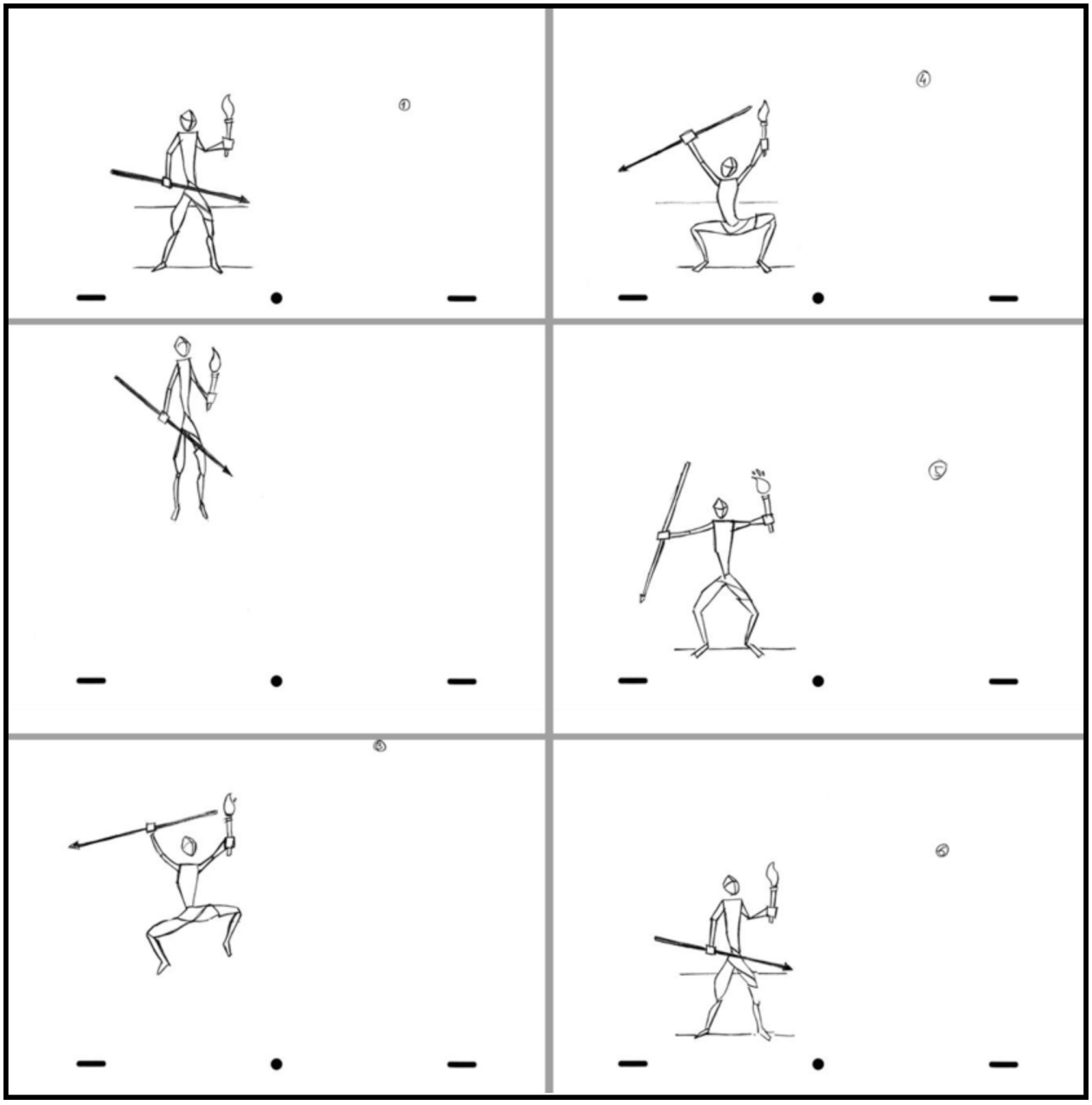


Fig. 2.26. Ciclo de salto de personaje para Altamira Game. Cazador

Fig. 2.27. Página web del programa de trazado vectorial Autotracer. www.autotracer.org

3. ANIMACIÓN BÁSICA FRAME A FRAME CON KRITA

María de Iracheta Martín
Identificador ORCID: 0000-0002-0246-0727

El proyecto del que forma parte esta propuesta es la creación de un videojuego a partir de software libre. Para conseguir el objetivo, cada miembro del equipo ha realizado una pieza clave necesaria para que el engranaje final funcionara. Siguiendo con el trabajo de años anteriores[6], se ha seguido profundizando en el programa de dibujo y pintura Krita para la realización del diseño de personajes. El nuevo paso que se da en esta ocasión, es dotar de movimiento a dichos personajes mediante el panel de animación del mismo.

Antes de llegar a ese punto, conviene señalar, que el concept art de este proyecto, no buscaba obtener personajes o escenarios de un mundo imaginado, al contrario, se buscaba ser lo mas fiel posible al diseño original, es decir, a las pinturas y relieves de las cuevas de Altamira.

Tras la fase de documentación, mediante Krita, se unificó el estilo gráfico de los personajes que iban a aparecer en el videojuego y que habían sido creados inicialmente por diferentes miembros del equipo con otros softwares libres. Como eran programas de dibujo vectorial, la aplicación del color en tintas planas, no se asemejaba mucho a la técnica original de las cuevas de Altamira, en la que los pigmentos ocres, rojizos y negros, se aplicaron con las manos aprovechando los relieves naturales del terreno creando diferentes matices.

Por ejemplo, el protagonista del juego, el personaje humano que va en busca del fuego a través de galerías y pasarelas rocosas, fue realizado con Inkscape a línea. Con el Krita, se aplicó color, volumen y se adaptó ligeramente la anatomía para que la animación fuera más coherente. El resultado se observa a continuación (figura 3.1).

Fig. 3.1. Diseño a línea de Borja Jaume y adaptación para animación y color de María de Iracheta

Del mismo modo, el resto de personajes fueron pasando por el proceso de unificación de estilo con Krita, dado que este programa ofrece varios pinceles, como el de cerdas duras, así como la esponja, que proporcionaban una textura que se adaptaba más a la pintura originaria. Para la gama cromática y la línea, se hizo *photobasing*, es decir, se puso de referencia en la capa 0 la imagen del personaje obtenida de la página del museo de Altamira para trabajar sobre ella en otras capas. De este modo, se pudo seleccionar con el cuentagotas los tonos exactos, así como la línea (figura 3.2)

[6] Ver: http://softwarelibrebellasartes.blogspot.com/p/krita.html (Consultado 14/01/2020)
https://softwarelibrebellasartes.wordpress.com/diseno-de-personajes-con-krita/ (Consultado 14/01/2020)

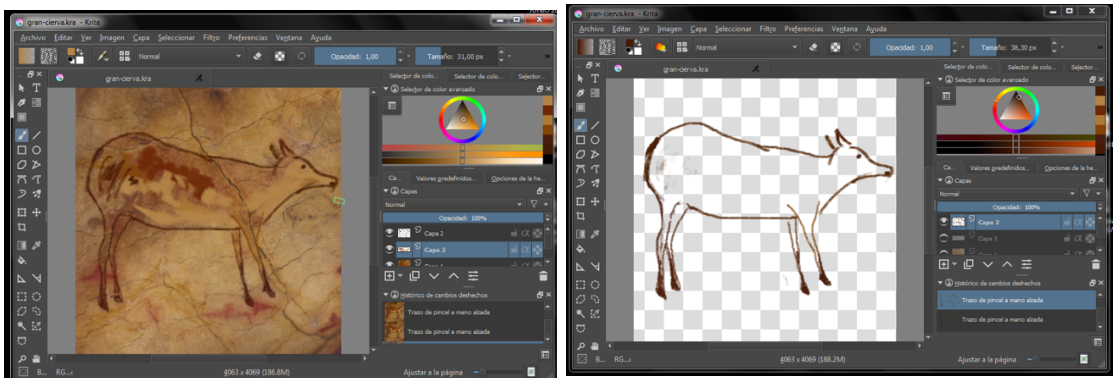
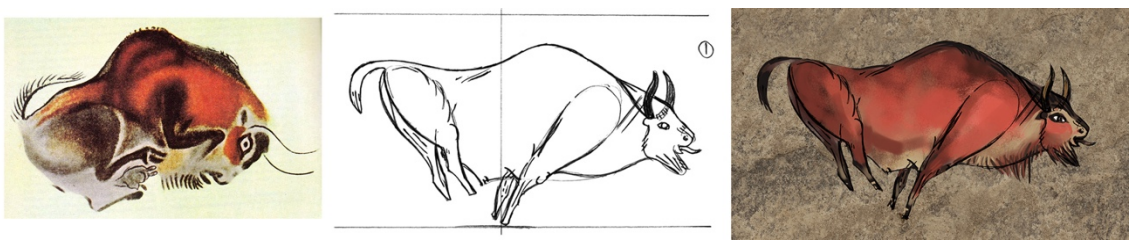


Fig. 3.2. Interfaz del programa Krita en el que se aprecia el proceso de obtención de línea y color mediante el uso de capas, cuentagotas y pinceles

Es importante mencionar, que dentro del proceso de trabajo de un videojuego, como es este caso, suele pasar que hay que rediseñar personajes en varias ocasiones. Un ejemplo se observa en la figura 3, en la que aparece la imagen de la cierva original de la cueva, después la cierva intentando ser lo más fiel posible a la primera, y finalmente la cierva mucho mas estilizada para que la animación quedara mas esbelta. También influyó en la tonalidad el modo de fusión que se iba a utilizar respecto al fondo rocoso. Como la idea era que pareciese que las pinturas cobraban vida, no una cierva real por delante de la pared, se utilizó el modo de fusión de multiplicar y se rebajó la opacidad consiguiendo que pareciera pintada sobre la roca. El inconveniente de esta acción es que muchos de los tonos se perdían al fundirse con la roca. Es un ejemplo de toma de decisión a lo largo del proceso de trabajo. Tras debatirlo, se decidió optar por perder tonalidad y mantener la idea de pintura con vida. De este modo, se establecía la diferencia entre el cazador, que es mas corpóreo al estar en lo que sería el plano real de la cueva, y animales.

Fig. 3.3. Fotografía de Altamira y posteriores adaptaciones de María de Iracheta

El bisonte no supuso tanto problema porque la pintura rupestre tiene unos tonos y trazos mas definidos (figura 3.4). Sin embargo, no fue así con el caballo, del cual no había casi referencias de color, por lo que se optó por usar matices más parecidos a la cierva pero con una personalidad propia (figura 3.5).



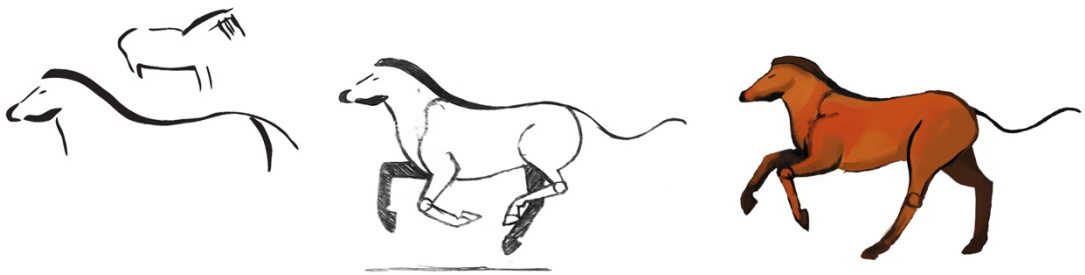


Fig. 3.4 y 3.5. (De izquierda a derecha) Imágenes originales de Altamira, dibujos de la animación de Carmen Pérez y entintado de María de Iracheta

Una vez terminados los personajes con una estética parecida y preparados para animar, cada uno con su desglose de poses en los ciclos de carrera, era el momento de montar la secuencia de fotogramas.

En el ejemplo que se explicará a continuación, se realizó un boceto a mano de la animación de carrera del cazador que sirvió de base para la animación con Krita. Esta imagen (figura 3.6), se colocó como capa 0 y se creó una capa por encima que sería donde se realizara la animación fotograma clave a fotograma clave.

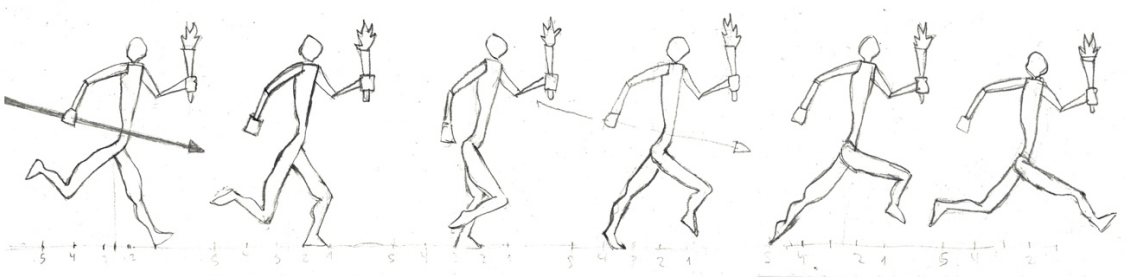


Fig. 3.6. Boceto a lápiz ciclo de carrera cazador pierna izquierda. María de Iracheta

Comenzamos abriendo el programa y creando un nuevo documento que le daremos las medidas mas utilizadas en animación 1920x1080 y la resolución será de 72ppp. Preparamos el espacio de trabajo para animar. Dependiendo de la versión del programa se encontrará en Ventana: espacio de trabajo: Animación, o en la esquina superior derecha en un cuadradito se puede también elegir este modo. En Preferencias activamos el Panel: animación. Hecho esto, aparecerá en la zona inferior del programa, la línea de tiempo donde crearemos los fotogramas claves, y el panel de animación donde controlar las acciones básicas de la animación: cuando comienza y termina, la velocidad de fotogramas, dar a reproducir la película y el papel cebolla (figura 3.7).

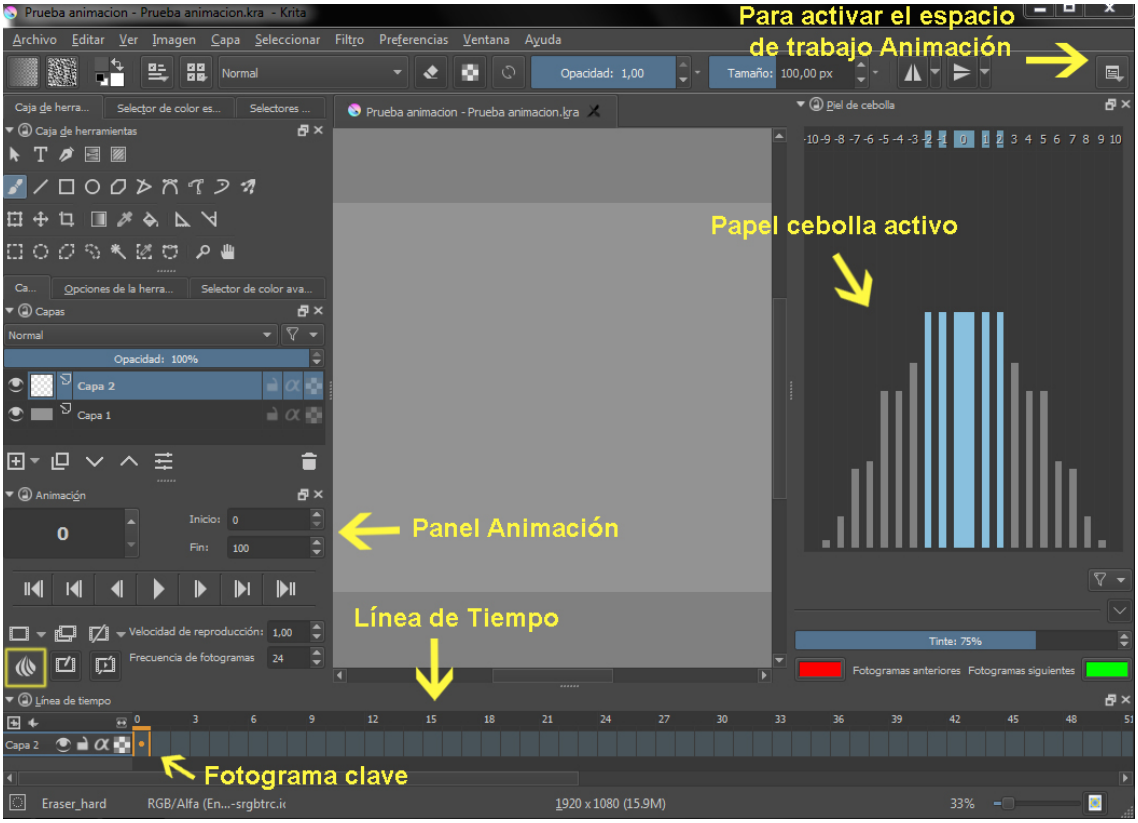


Fig. 3.7

Para poder ver las capas que se necesitan en la línea de tiempo, tenemos que seleccionar la capa y con botón derecho activar Mostrar en la línea de tiempo. Así es como se ha podido calcar las poses claves del boceto a lápiz de la animación.

A continuación se realizan los siguientes pasos: se crea fotograma clave (por defecto el fotograma 1 ya es clave) y dibujamos; se selecciona el fotograma donde queremos colocar el siguiente fotograma clave y se da botón derecho Nuevo fotograma y dibujamos; volvemos a seleccionar el siguiente fotograma donde irá un clave, creamos y dibujamos, y así sucesivamente hasta terminar la animación.

Se puede realizar el dibujo integro desde cero o como en este caso calcando cada pose del boceto a mano. El boceto se ha tenido que ir desplazando hacia la izquierda en cada pose para hacer coincidir en el mismo eje el personaje y que la animación se produzca en el mismo sitio. Si tuviéramos un fondo lo colocaríamos en una capa por debajo y fotograma a fotograma tendríamos que irlo desplazando hacia la derecha y así conseguiríamos el efecto conocido como *parallax*, con el que parece que el personaje, sin moverse del sitio, avanza en el espacio. Si activamos el papel cebolla en Preferencias: Piel de cebolla, podemos elegir cuantos fotogramas por delante y por detrás del fotograma activo vemos y así poder calcar también zonas que se repiten y modificar la parte que se va moviendo. El panel de piel de cebolla se puede ocultar haciendo clic en el icono de la cebolla del panel de animación. Para activarlo en la línea de tiempo hay que presionar un icono que muestra una bombilla y que aparece en la capa de animación en la que estamos trabajando. También es útil que se puede cambiar el color de la línea del dibujo de los fotogramas que salen en transparencia así como su opacidad aumentando o disminuyendo las líneas azules que aparecen en el panel de piel de cebolla (figura 3.8). Del mismo modo, en la línea de tiempo, se pueden poner también colores a los diferentes tipos de fotograma, y así poder seleccionar lo que interese en cada momento.

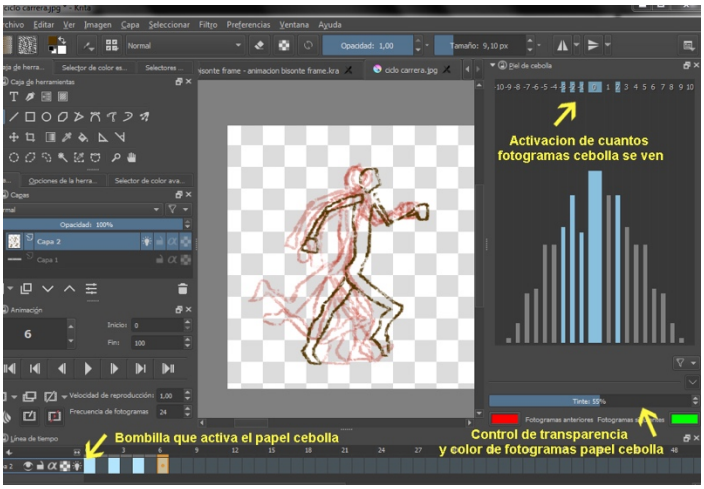


Fig. 3.8

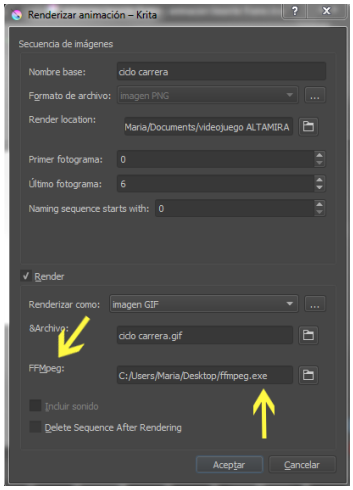


Fig. 3.9

Una vez dibujados todos los fotogramas de la animación, ya sean fotogramas clave como intercalaciones (dibujos de transición entre dos fotogramas clave) hay que decidir el *timing,* es decir, la duración de cada fotograma, lo que va a dar el ritmo a la película. En este caso se ha utilizado la frecuencia de animación clásica de 24 fotogramas por segundo y se ha establecido la secuencia de fotogramas clave a doses, es decir, cada dos fotogramas. Si se pone a unos, iría demasiado rápido y a treses podría funcionar por ejemplo en la parte en la que el personaje está con la zancada en el aire, prolongándola ligeramente para volver a doses en el fotograma ya de contacto del pie con el suelo. Lo ideal es que se vaya probando lo que funciona mejor para cada personaje.

Por último necesitamos exportar la animación a un formato de vídeo o GIF. para poder visualizarlo como película. Para exportar necesitamos descargar el software libre FFmpeg[7] porque Krita no puede por si solo. Tendremos que ir a Archivo: Renderizar animación y sale el siguiente panel (figura 3.9) en el cual, podremos seleccionar el formato de salida video o GIF. En el apartado de FFMep habrá que seleccionar el archivo ejecutable .exe de la descarga realizada en la página del programa y una vez que aceptemos ya tendremos la animación lista para verla en movimiento.

Como se puede comprobar, Krita no sólo es un buen software para diseño de personajes y entornos en videojuegos, sino que ofrece al alumno o profesional una herramienta de animación bastante completa si se trabaja con animación fotograma a fotograma. El papel cebolla aporta un buen ritmo de trabajo a través de su sistema de colores y opacidades, muy útil si se trabaja con varios elementos a la vez en la animación. Se concluye afirmando que la unión de Krita, con otros programas de software libre, es una alternativa viable tanto para la industria de la animación como de los videojuegos.

[7] https://www.ffmpeg.org/ o https://ffmpeg.zeranoe.com/builds/ (ambas consultadas el 15/01/2020)

4. CREACIÓN DE PERSONAJE Y PROPS CON INKSCAPE

Borja Jaume Pérez
Identificador ORCID: 0000-0001-7156-9059

Una vez decidido el argumento del videojuego, se comenzó a diseñar el concept art de los personajes con la premisa de utilizar herramientas de software libre durante todo el proceso de trabajo. Tanto el protagonista, un cazador/recolector del Paleolítico, como la antorcha que éste portaría y los elementos que iban a formar parte del escenario (*props*), fueron diseñados con el software libre de gráficos vectoriales Inkscape.

Una de las pautas a la hora de abordar el diseño consistía en tratar de acercarse lo más posible a las pinturas del arte rupestre, como las que se pueden encontrar en las cuevas de Altamira; además de otro tipo de elementos que podían servir de inspiración para los props, como las puntas de sílex. Inkscape resultó ser una herramienta ideal para este cometido ya que, tanto los dibujos de las cuevas como los materiales allí encontrados, sugerían diseños con contornos sencillos y marcados que podían ser recreados de manera adecuada a través de las formas, trazos y transformaciones disponibles en el software.
La primera parte consistió en obtener documentación sobre las figuras humanas pintadas en las cuevas del Paleolítico.

Fig. 4.1. Imágenes de inspiración para el personaje principal. (Izquierda) reproducción de Juan López del Toro de los yacimientos Prehistóricos del *Abrigo de Mulano* (Murcia). (Derecha) pintura rupestre del yacimiento de *La Sarga*, Alcoy, Alicante

También se llevó a cabo una labor de búsqueda de referencias en relación a las lámparas que usaban para poder pintar en las cuevas y las puntas de sílex que utilizaban como armas o herramientas. En este punto, fue especialmente útil el asesoramiento del profesor Pedro Saura, director y autor, junto con Matilde Múzquiz, de la réplica del techo polícromo de la cueva de Altamira, conocida como Neocueva, instalada en el Museo de Altamira (Cantabria), ya que nos permitió tener una idea más precisa sobre la manera en que estos instrumentos fueron utilizados, permitiéndonos así tener un grado de rigurosidad adecuado para el proyecto.

Fig. 4.2. Detalle de punta de sílex e ilustración de Arturo Asensio para la exposición *La mirada del Paleolítico*

Una vez obtenida la documentación necesaria se empezaron a diseñar los primeros bocetos con Inkscape, siempre tratando de crear formas sencillas que recrearan las pinturas originales. Para ello se recurrió a las herramientas de dibujo a mano alzada y curvas Bézier.

Con el fin de lograr que el personaje tuviera más movilidad y poder crear un *idle* más dinámico, se decidió alargar sus extremidades. Además se le añadió una lanza y una antorcha que, posteriormente, se cambió por el cuenco de piedra, mucho más acorde con las lámparas usadas en Altamira.
El siguiente paso consistió en hacer pruebas de color para el personaje en base a fondos y texturas de piedra y tonos cobrizos.

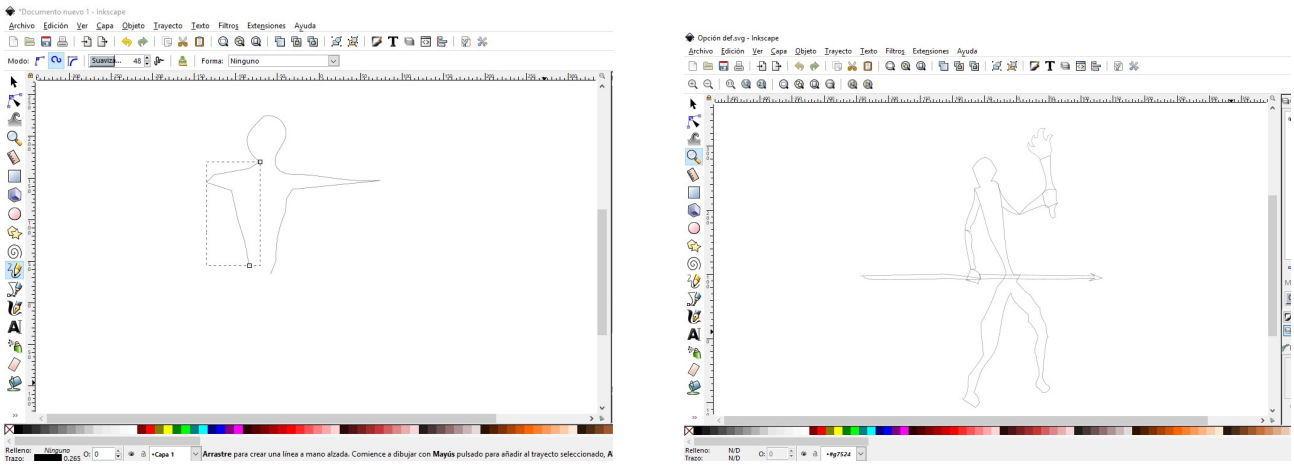


Fig. 4.3. Diseño del cazador con Inkscape

Para realizarlo, se llevaron a cabo los siguientes pasos en Inkscape:

1. Añadir una imagen de textura de piedra que resultara adecuada
2. Situar sobre esa textura el dibujo u objeto creado como una capa nueva
3. Ir a ‘Objeto’ - ‘Recorte’ - ‘Aplicar’

De esta forma, el fondo de piedra cubría al personaje dando la textura y el color que se buscaba.

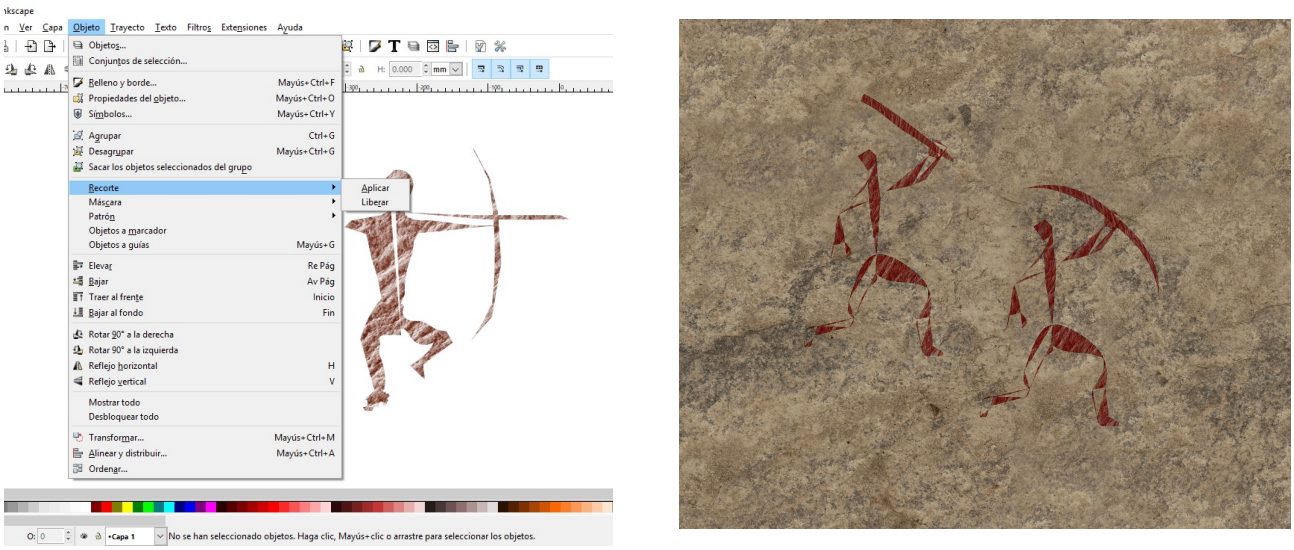


Fig. 4.4. Aplicando textura al personaje mediante Inkscape

Una vez elegido el diseño de personaje definitivo, la profesora María de Iracheta modificó el color con el *Software* Krita para darle volumen. Este proceso fue necesario debido a que Inkscape no es un programa de pintura digital como tal y la gama de pinceles y colores que encontramos en Krita es mucho más rica. También se suavizó ligeramente la anatomía y finalmente se dividió en piezas.
Aprovechando la gama de colores utilizados en Krita para el coloreado del personaje, se llevó a cabo la tarea de, no solo diseñar el cuenco y la punta de silex en Inkscape, sino también de pintarlo. Para ello, Inkscape dispone del 'panel de relleno y borde' en la parte derecha de la interfaz que permite, mediante diferentes modos de color (RGB, HSL, CMYK, rueda y CMS), hacer modificaciones de tono, brillo y saturación. Además de tener opciones para realizar diferentes degradados como: degradado lineal, radial, de rejilla etc. Estas herramientas sirvieron por tanto para lograr transiciones de color y sombreados suaves.

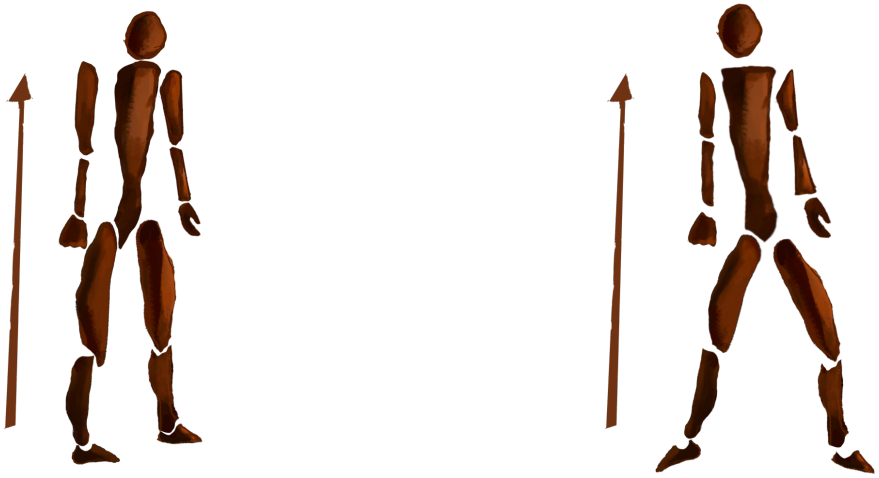


Fig. 4.5. Personaje final retocado en Krita por María de Iracheta

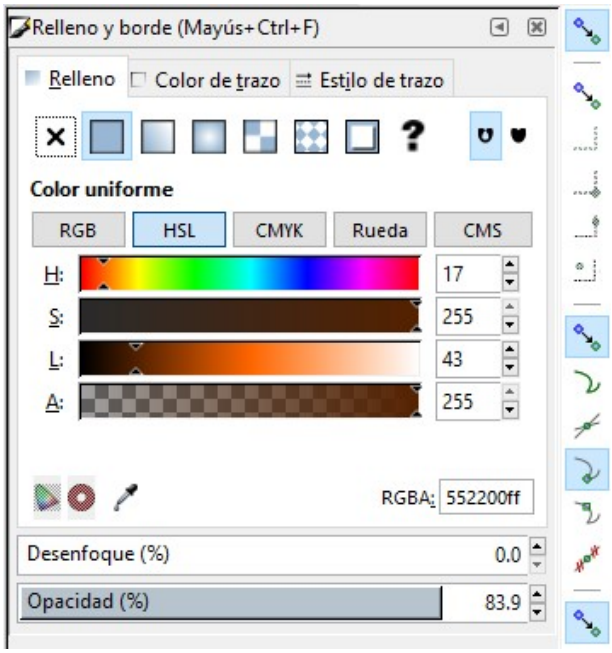


Fig. 4.6. Panel de relleno y borde

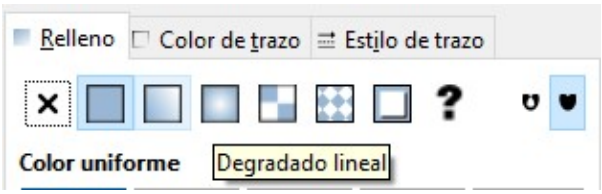


Fig. 4.7. Detalle de los diferentes modos de degradado.

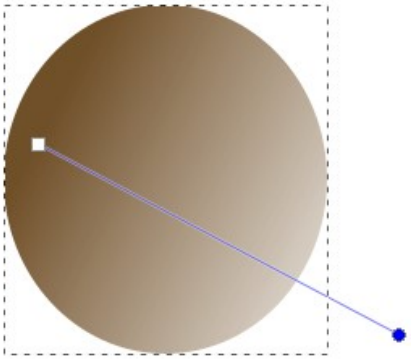


Fig. 4.8. Ejemplo de degradado lineal

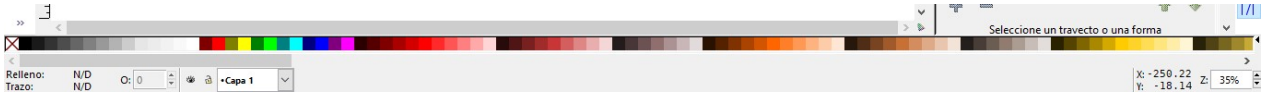


Fig. 4.9. Barra de colores de Inkscape

Proceso de creación del cuenco y la punta de sílex en Inkscape

Lo primero que haremos es abrir un documento nuevo. Nos vamos a 'Archivo', 'Propiedades del documento' y en la opción 'borde' deseleccionamos 'mostrar borde del papel', de esta manera tendremos el espacio de trabajo completamente vacío.

Una vez hecho esto seleccionamos el color que nos interese en la barra de colores situada en la parte de abajo. En este caso será el color marrón.

Ahora seleccionamos la herramienta 'Crear elipses' de la barra de herramientas y creamos dos elipses del mismo tamaño. Después vamos a la herramienta 'Crear rectángulos' y creamos un rectángulo con un ancho mayor que las elipses. Situamos el rectángulo encima de una de las elipses, justo ocupando la mitad superior. Nos vamos a la barra de menús (arriba) y en la ventana 'Trayecto' y pulsamos la opción 'Diferencia' para realizar un corte en la elipse aprovechando la forma del rectángulo. Colocamos la segunda elipse sobre la primera que había quedado recortada de tal forma que quede una forma que pueda recordar a un cuenco. Finalmente creamos otra elipse más pequeña para la parte del orificio del cuenco.

Lo siguiente será seleccionar el objeto creado mediante las dos elipses ir a 'Trayecto', 'Unión' para que se conviertan en un único objeto.

Posteriormente se empiezan a hacer las pruebas de color mediante degradados. Vamos al objeto cuenco, botón derecho y lo duplicamos. Lo oscurecemos eligiendo un tono marrón más oscuro. Después vamos a la barra de degradados, seleccionamos degradado lineal y creamos tantos degradados como consideremos oportuno.

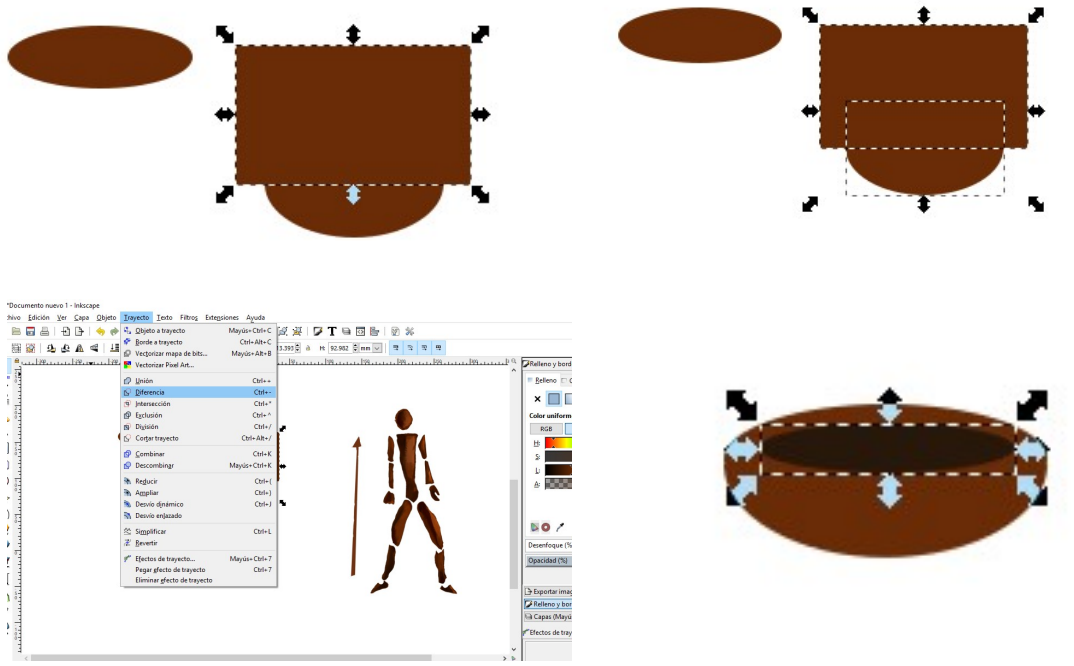


Fig. 4.10. Proceso de creación del cuenco

Después seleccionamos el objeto cuenco y el objeto orificio y vamos a ‘Objeto’, ‘Agrupar’ para que los dos elementos funcionen como uno solo.

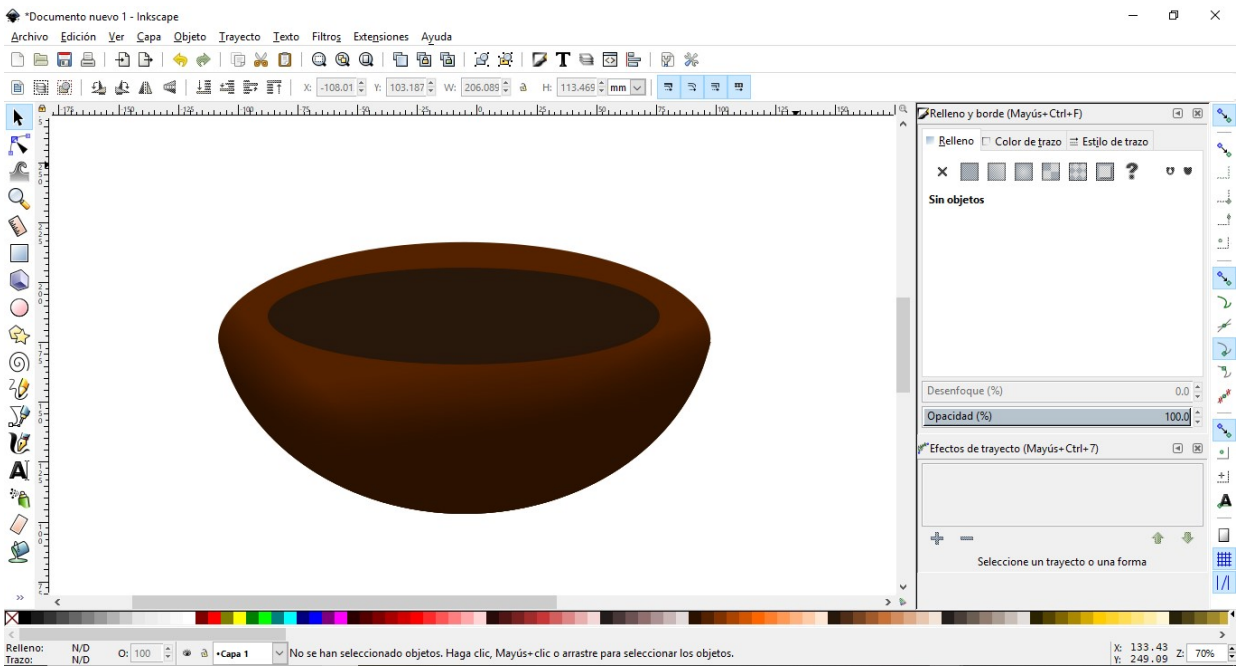


Fig. 4.11. Diseño de cuenco mediante degradados

Se pueden probar otros colores y degradados en función de lo que nos interese. En este caso se aclaró un poco los tonos marrones en la siguiente versión.

En cuanto a la punta de sílex el proceso empezó con la selección de unas imágenes de referencia para crear la pieza.
Abrimos un nuevo archivo y arrastramos la imagen de referencia dentro. Después creamos una figura similar mediante curvas Bézier y la rellenamos del color que creamos conveniente mediante la herramienta 'Bote de pintura'. Una vez tenemos nuestra figura creada, la situamos sobre la imagen de referencia para aprovechar la textura y generar la sensación de piedra. Esto lo hacemos desde 'Objeto' (barra de menús), 'Máscara'.

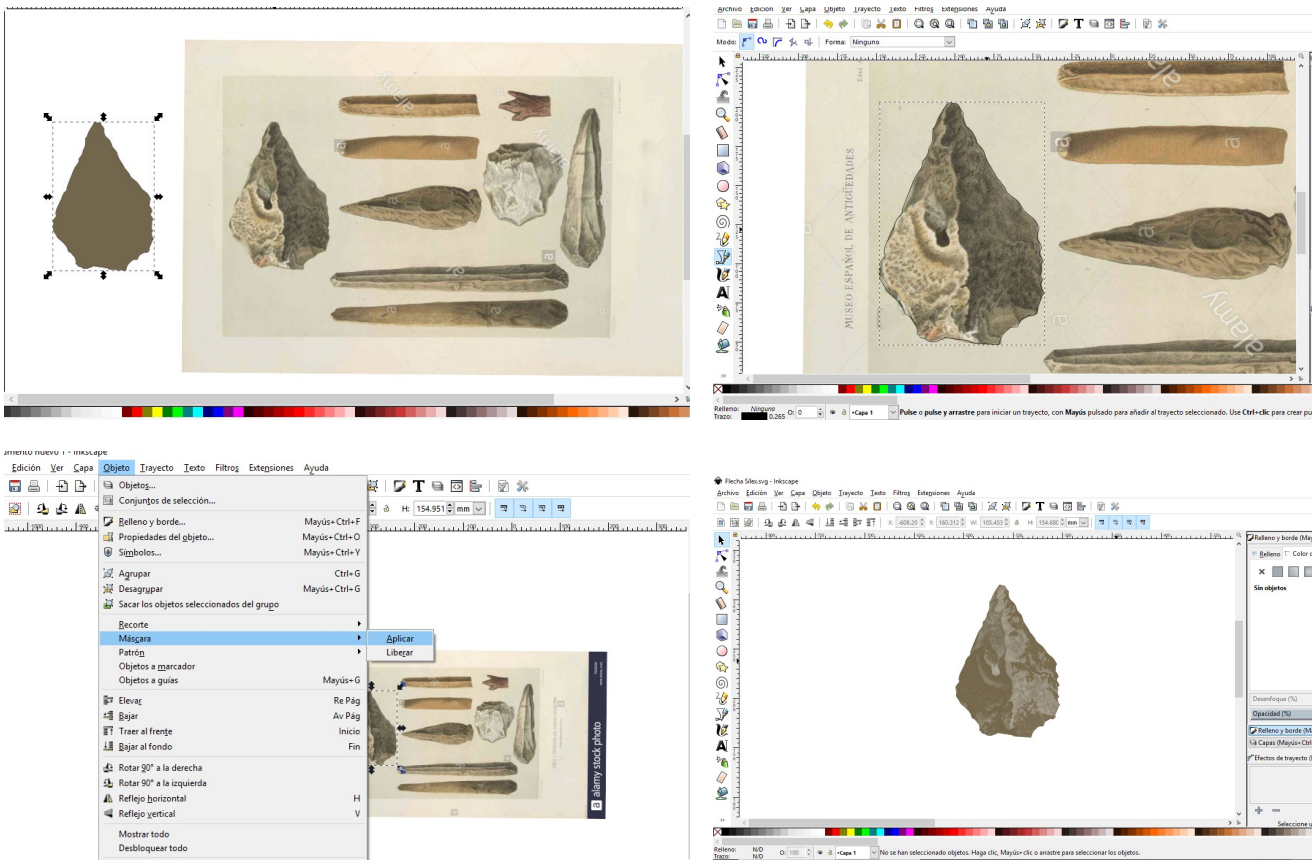


Fig. 4.12. Proceso de creación de la punta de sílex en Inkscape

Para terminar realizamos otra versión mediante la misma técnica.

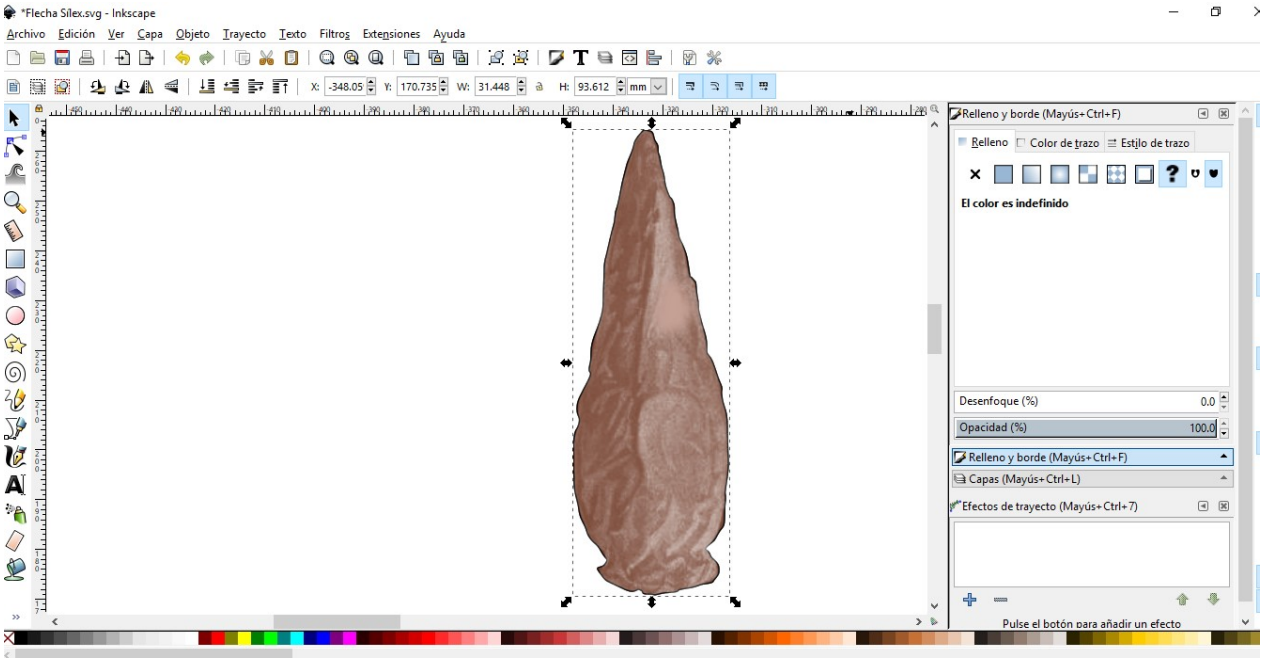


Fig. 4.13. Segunda versión de la punta de sílex en Inkscape

5. CREACIÓN DE ELEMENTOS 3D CON BLENDER

Darío Lanza Vidal
Identificador ORCID: 0000-0003-0475-2114

Aunque el proyecto se plantea como un videojuego en un espacio 2D, hemos querido incorporar ciertos elementos tridimensionales cuyo propósito es añadir una cierta corporeidad y presencia física al entorno. En concreto la pared de la cueva, con su característica superficie irregular y abultada, constituye un elemento muy interesante para ser construido tridimensionalmente y que de esta manera reaccione a la iluminación incrementando la sensación de entorno cavernoso. Otros elementos susceptibles de ser construidos en 3D han sido las plataformas que, por su naturaleza rocosa, se relacionan más con la fisicidad tridimensional de la pared que con la planitud de los personajes dibujados en su superficie.
Para la creación de estos elementos en 3D hemos escogido Blender por constituir una poderosa plataforma libre de creación tridimensional y Krita como software gráfico para el trabajo de texturas.

Modelado y texturizado de la pared de la cueva

La pared de la cueva presenta en este proyecto un papel protagonista como espacio en el que se desarrolla el relato del videojuego y por ello hemos prestado especial atención a sus características aspectuales, tanto en su construcción tridimensional como en las texturas de su superficie. De cara a evitar sobrecargar la memoria con excesiva geometría y texturas hemos planteado la pared como un elemento modular que pueda repetirse tantas veces como sea necesario para abarcar todo el desarrollo longitudinal que exijan las dimensiones del nivel. Para ello es imprescindible que tanto la geometría de la pared como su textura presenten continuidad horizontal y vertical, el concepto conocido como *tileable*, de modo que puedan repetirse indefinidamente sin que se muestren líneas de sutura que evidenciarían la naturaleza modular y repetitiva del entorno. Con esta premisa hemos creado la pared a partir de un plano cuadrado subdividido de 5x5m, que hemos deformado empleando las herramientas de escultura de Blender. Para provocar el tipo de deformación característico de la erosión en la roca caliza han resultado esenciales los pinceles de escultura *Drag*, *Inflate* y *Smooth*, restringiendo la deformación a las áreas centrales del plano de modo que los márgenes se mantuvieran intactos y de esta manera se preserva el imperativo de continuidad geométrica (figura 5.1).

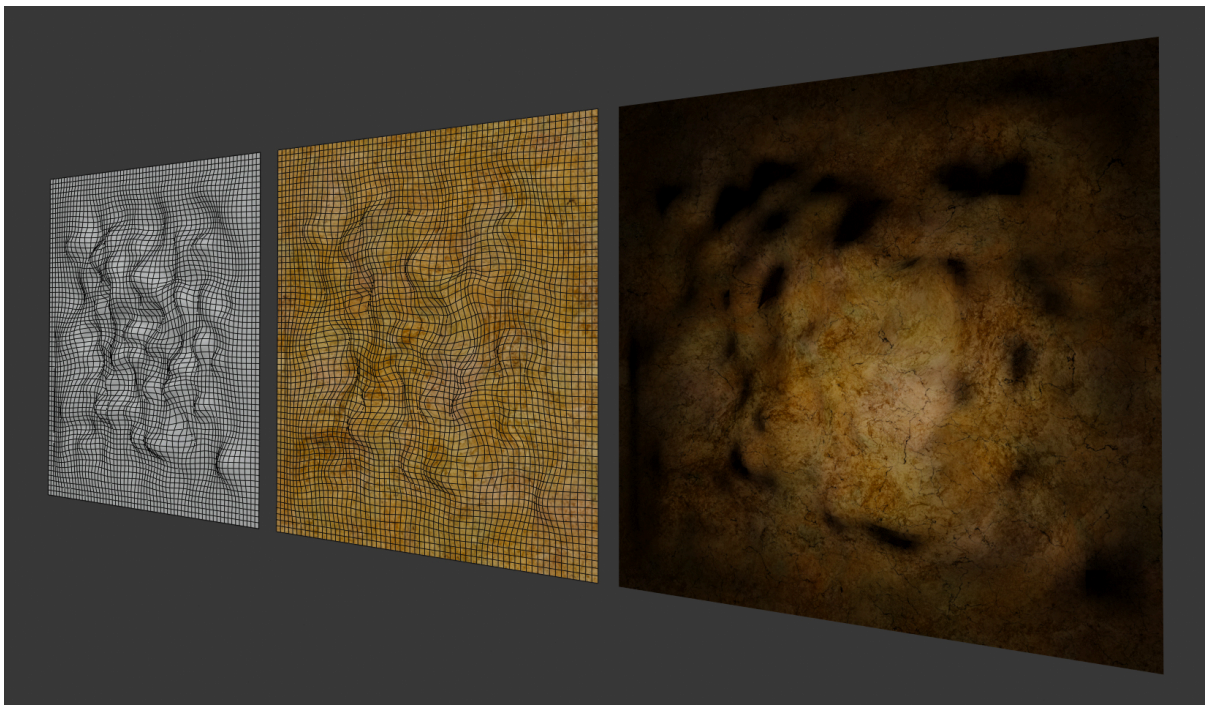


Fig. 5.1. Modelo 3D del módulo cuadrado de pared (izquierda), con textura (centro) y con iluminación (derecha)

Para la textura procedimos de la misma manera, diseñando una textura cuadrada con continuidad horizontal y vertical de manera que no mostrase líneas de sutura al ser repetida a lo largo de todo el nivel. Utilizamos una

textura de roca gratuita[8], y trabajamos tanto su mapa de normales (para simular microrrelieve) como la textura de color en Krita cuidando la mencionada continuidad de los márgenes. Dado que la superficie de la cueva de Altamira presenta multitud de grietas producto de procesos erosivos, creamos una textura procedural en Blender a partir de un procedural de *Voronoi* con el que conseguimos producir un patrón de grietas convincente que incorporamos a la textura de color (figura 5.2).

Fig. 5.2. Mapa de normales (izquierda), textura de color (centro) y patrón procedural de grietas generada en Blender (derecha)

Modelado de la plataformas de piedra

De cara a relacionar matéricamente las plataformas con la pared de la cueva, decidimos que estas también fueran modeladas en 3D, diferenciándose así por su tridimensionalidad el contexto de lo rocoso de los personajes y animales planos dibujados. Para la creación de estas plataformas rocosas resultó muy útil la primitiva *Rock Generator*, incorporada como add-on en Blender en su versión 2.8. También en este caso se buscó la máxima modularidad para tratar de proporcionar la mayor variabilidad visual con un número mínimo de elementos. Para ello se modeló una colección de plataformas que pudieran utilizarse en distintas orientaciones y escalas, de modo que su reutilización no resultase visualmente repetitiva (figura 5.3). También se limitó el número de polígonos a 1000 en cada objeto, de modo que se redujesen los requisitos gráficos necesarios y de esta manera se garantizase el funcionamiento fluido del videojuego en cualquier plataforma y navegador web.

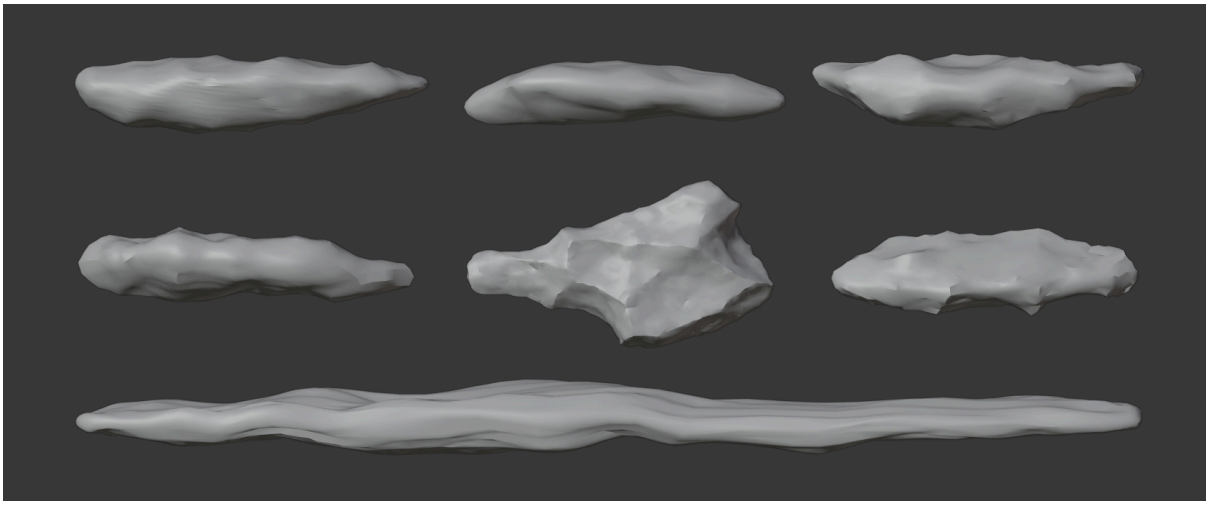


Fig. 5.3. Plataformas rocosas modeladas en Blender. Su forma se ha diseñado buscando la máxima versatilidad de cara a que su reutilización no resulte visualmente repetitiva

[8] En concreto utilizamos la textura gratuita *Rough Plaster 03* del portal TextureHaven (www.texturehaven.com)

6. MONTAJE DEL VIDEOJUEGO EN UNITY

Darío Lanza Vidal
Identificador ORCID: 0000-0003-0475-2114

En el presente proyecto hemos escogido Unity como plataforma para el desarrollo del videojuego, el cual, no siendo un software libre, es gratuito y se encuentra ampliamente extendido como estándar en el sector. El primer paso en la construcción del videojuego es la creación en Unity de un proyecto 2D y la importación del modelo que servirá como pared de la cueva y fondo del nivel. Todos los elementos que importemos en Unity se guardarán en la carpeta *Assets*, y para mejorar la organización de estos archivos crearemos subcarpetas destinadas a albergar las animaciones, los objetos 3D, las escenas, los scripts, los shaders, los sonidos y las texturas.

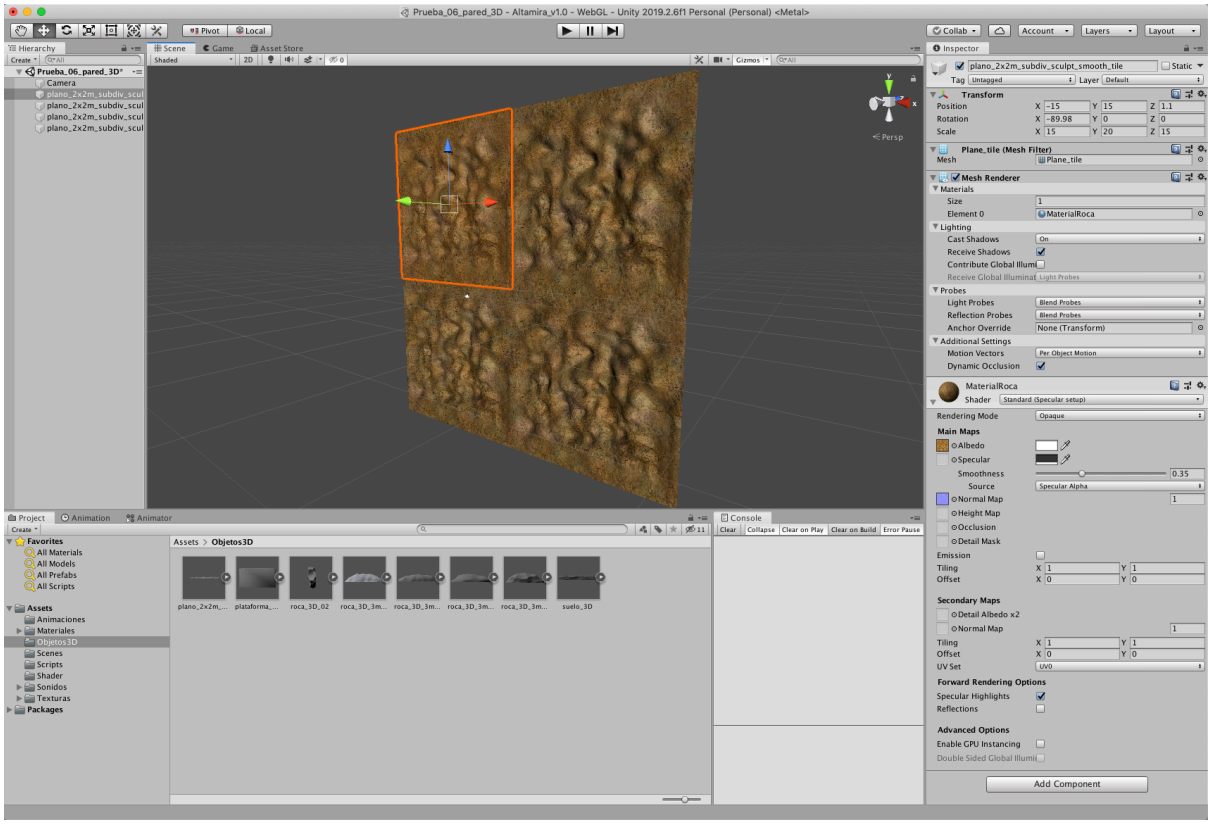


Fig. 6.1. Importación del módulo 3D de pared. La continuidad tanto del modelo como de la textura garantizan que la repetición no muestra líneas de sutura. A la derecha, descripción del material de la roca

Para la creación del material de la pared se ha utilizado un *shader Standard*, al que se ha aplicado la textura de color (Albedo) y el mapa de normales simulando el microrrelieve de la roca. Un ligero *Smoothness* (0.35) ha sido útil para simular la especularidad de la superficie húmeda de la cueva. Podemos comprobar que la continuidad geométrica del modelo 3D y de su textura consigue evitar la aparición de líneas de sutura cuando el módulo de pared se repite horizontal y verticalmente.

Uno de los elementos más habituales en un videojuego 2D de estas características son las **plataformas**. Para configurar cada plataforma en Unity importamos el modelo 3D procedente de Blender y le agregamos un colisionador –en nuestro caso hemos empleado un *Box Collider 2D* adaptado al tamaño del objeto–, activamos la opción *Used By Effector* y agregamos un *Platform Effector 2D*, que consigue que la plataforma sólo opere cuando el personaje cae sobre ella desde arriba, facilitando el salto desde abajo.

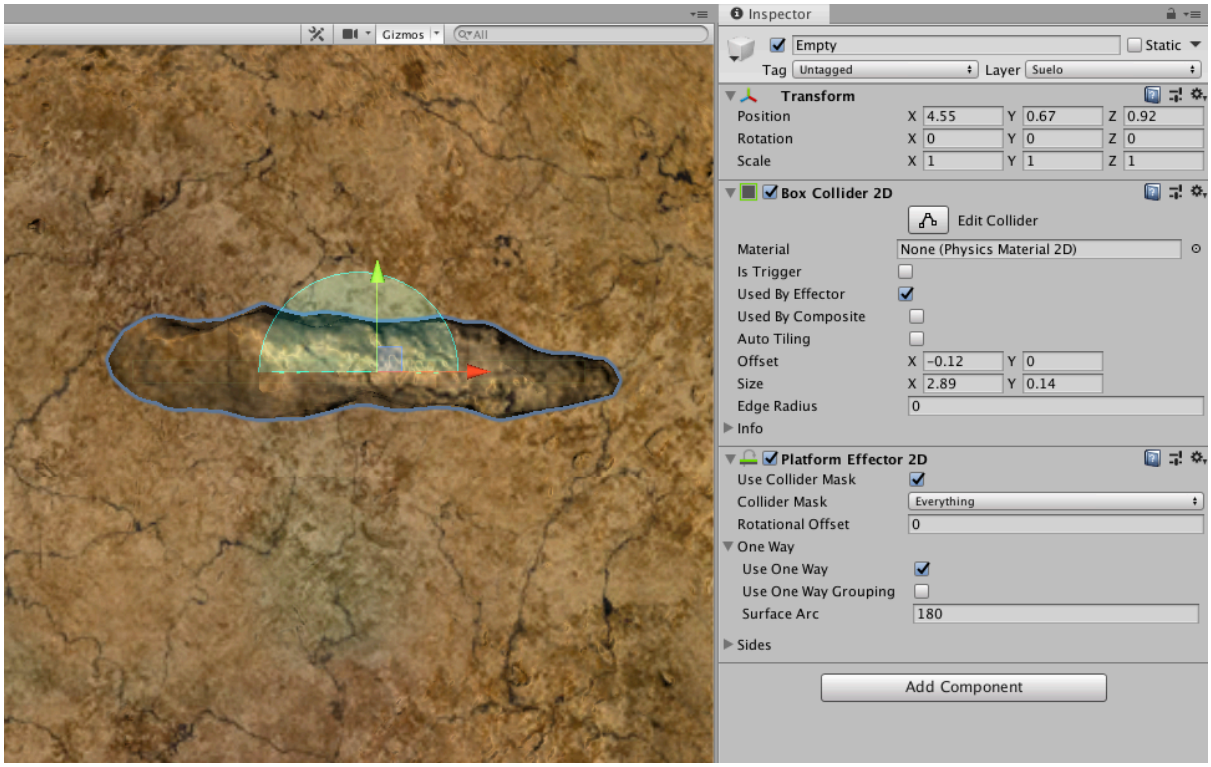


Fig. 6.2. Configuración de las plataformas. Colisionador y Platform Effector

Debido a la ubicación del videojuego en el interior de una cueva, utilizamos la **luz de las lámparas de aceite** como elemento no sólo estético sino también narrativo, haciendo que el agotamiento del combustible fuera atenuando la intensidad de la luz y con ello operara como un time-out, que obliga al jugador a recargar su lámpara con aquellas que se encuentra en su camino para evitar sumirse en la total oscuridad. El primer paso para recrear este elemento fue la creación del fuego, para lo que empleamos una simulación de partículas mediante un *Particle System* de Unity (figura 6.3).

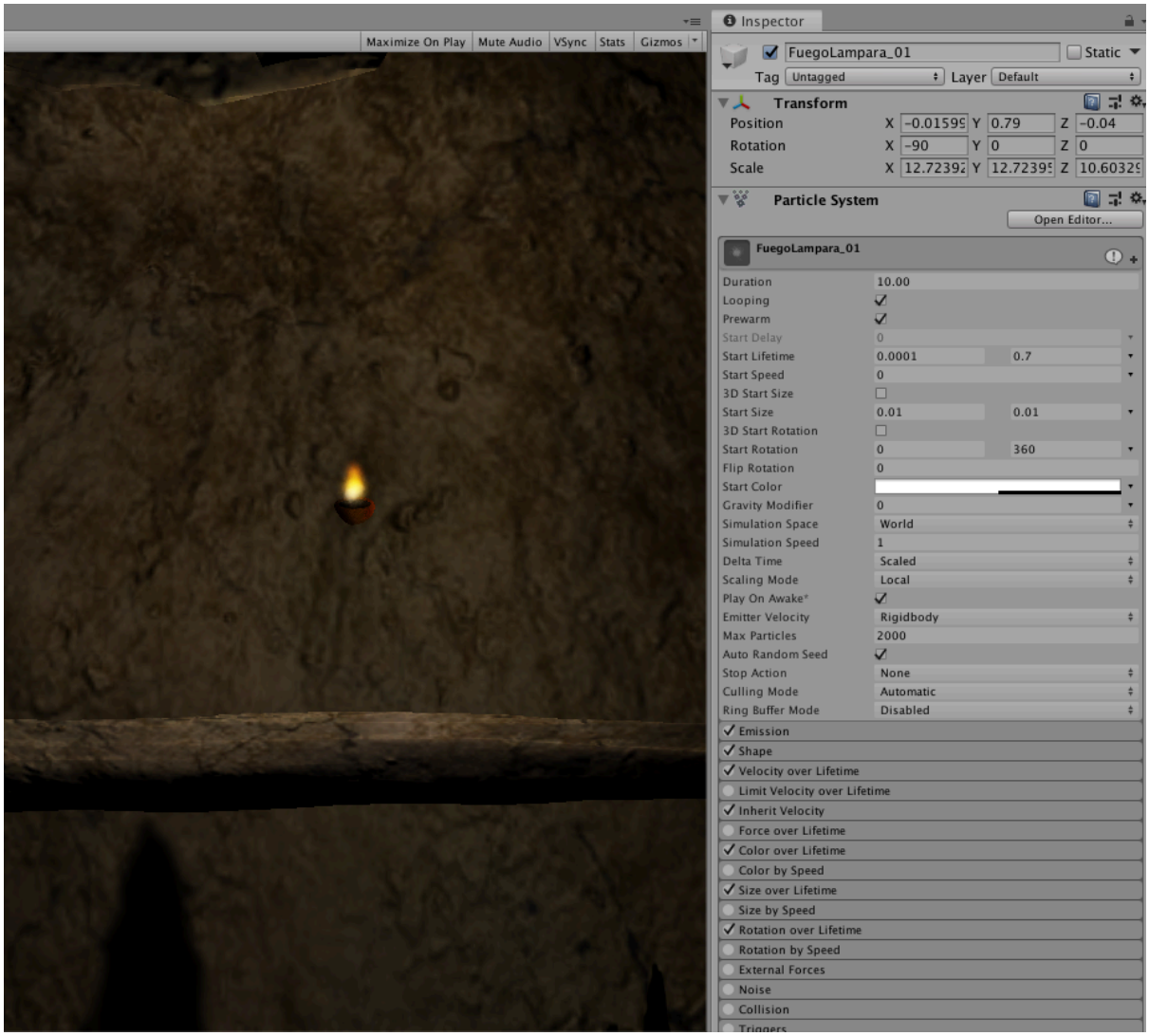


Fig. 6.3. Configuración del fuego empleando un Particle System

A este fuego, en principio un simple elemento bidimensional sin propiedades lumínicas, se le agregó una luz de punto, cuya intensidad se animó mediante key frames de forma oscilatoria para reproducir el característico efecto tembloroso de la luz de antorcha. Para que la luz de la lámpara se fuera consumiendo paulatinamente desde el momento en que el jugador la coge, fue necesario programar este comportamiento mediante un script. Dicho script (que reproducimos aquí en la figura 6.4) toma la intensidad actual de la luz dada y en cada fotograma del videojuego la reduce en una cantidad definida por el usuario.

```
// Script Intensidad Decreciente
// Autor: Darío Lanza
// Lenguaje: C#

// Script que modifica la intensidad de una luz y el tamaño de una llama, indicados por el usuario.
// Es imprescindible indicar la luz de destino y la llama sobre los que va a actuar.


using System;
using UnityEngine;
using System.Collections;

  public class IntensidadDecreciente : MonoBehaviour {

    public Light Luz;
    public GameObject Llama;
    public float tasaAtenuacion = 0.001f;
    public float llamaMinima = 2.0f;
```

```
    public float intensidadGameOver = 0.0f;

  void Start() {}

  public void Update(){
          Luz.intensity = Luz.intensity - tasaAtenuacion;
          if (Llama.gameObject.transform.localScale.x <= llamaMinima) {
             Llama.gameObject.transform.localScale = new Vector3(llamaMinima, llamaMinima,
llamaMinima);
             }
          else {
             Llama.gameObject.transform.localScale -= new Vector3(tasaAtenuacion*20, tasaAte-
nuacion*20, tasaAtenuacion*20);
             }
      }
}
```

Fig. 6.4. Script programa en C# para atenuar en el tiempo la intensidad de la luz que porta el jugador

De esta manera la lámpara del jugador irá perdiendo intensidad desde el momento en que este la coge, sumiendo gradualmente la escena en la oscuridad. De cara a que el jugador pueda recargar su lámpara y volver a disponer de iluminación, se disponen en la escena nuevas lámparas que, al ser cogidas, incrementan de nuevo la intensidad de su luz. Este efecto también fue programado mediante un script (figura 6.5).

```
// Script Modificar Luz
// Autor: Darío Lanza
// Lenguaje: C#

// Script que modifica la intensidad de una luz y el tamaño de una llama, indicados por el usuario.
// Es imprescindible indicar la luz de destino y la llama sobre los que va a actuar.


using System;
using UnityEngine;
using System.Collections;

public class ModificarLuz : MonoBehaviour {

   public Light Luz;
   public GameObject Llama;
   public float incrementoIntensidad = 0.5f;

   void Start() {}

   void OnTriggerEnter2D(Collider2D other)  {

      if(other.tag == "Player")  {
         Luz.intensity = Luz.intensity + incrementoIntensidad;
         Llama.gameObject.transform.localScale += new Vector3(12.5f, 12.5f, 10.5f);
      }
   }
}
```

Fig. 6.5. Script que vuelve a incrementar la intensidad de la lámpara del jugador

Para incrementar la inmersión en el ambiente cavernario que pretende el videojuego, **el sonido** ha desarrollado un papel destacado. Empleamos un archivo gratuito de sonido[9] para evocar las gotas que caen de las estalactitas. Editamos el sonido en Audacity, editor de audio open-source, en el que modificamos su duración, lo convertimos a Mono y le aplicamos un efecto de Reverberación para simular el eco del interior de la cueva (figura 6.6). Desde Audacity lo exportamos a MP3 y lo importamos en Blender, incorporándoselo a un objeto Audio Source.

[9] *Water Drops Sound*, creado por Daniel Simion. Disponible en http://soundbible.com/2186-Water-Drops.html

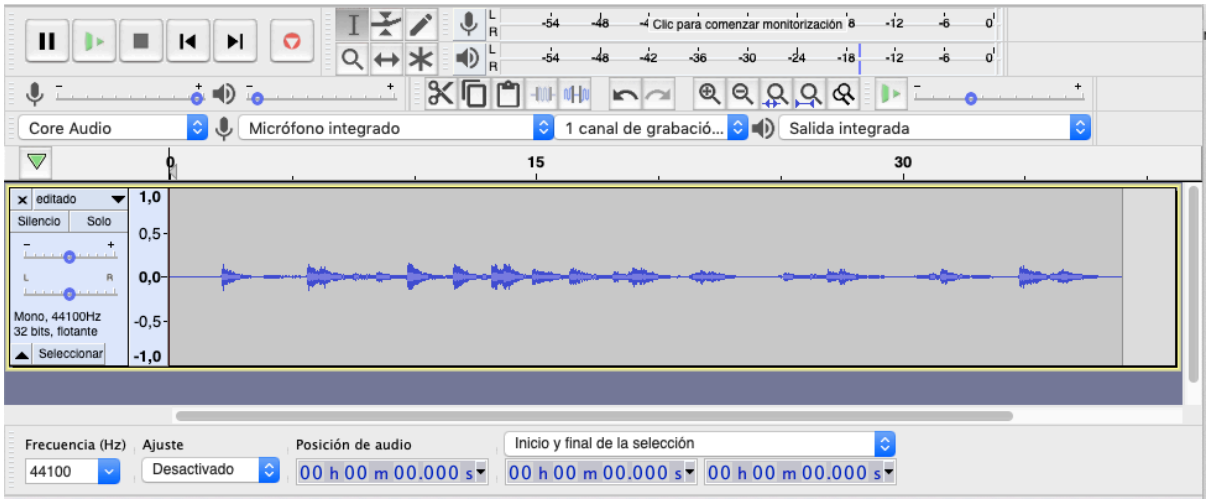


Fig. 6.6. Editando el sonido de gotas en Audacity

De cara a añadir una sensación de espacio en la percepción de este sonido, se aplicó a este Audio Source un script que reduce el volumen del audio en función de la distancia del cazador al propio objeto Audio Source, de modo que el jugador percibe que las gotas suenan desde un lugar en concreto y se atenúan al alejarse, produciendo una sensación más realista e inmersiva.
También se utilizaron sonidos gratuitos para el crepitar del fuego[10], el encendido de la lámpara[11] y la estampida del bisonte[12], editados igualmente en Audacity, además de una colección de sonidos creados por el diseñador de sonido Yago Cordero Domenech[13] para este proyecto.

El cazador, que encarna al jugador, es el personaje protagonista del videojuego. Su diseño, como describimos anteriormente, fue realizado en Inkscape, y desde Krita se exportaron todos sus miembros por separado en forma de PNG transparente, de modo que cada parte del cuerpo pudiera ser animado en Unity de forma independiente. Una vez importado el PNG en Unity y separados sus miembros utilizando el *Sprite Editor*, procedimos a crear las animaciones *idle*[14], carrera, salto y aterriza, siguiendo los diseños de animación creadas por Carmen Pérez, María de Iracheta y Borja Jaume. En Unity utilizamos el panel *Animation* para crear dichos ciclos mediante key frames (figura 6.7).

[10] *Campfire*. Creado por Cagan Celik. Disponible en https://freesound.org/people/CaganCelik/sounds/433783/
[11] *Match Ignite*. Creado por Dean Raul DiArchangeli. Disponible en https://freesound.org/people/Dean-Raul_DiArchangeli/sounds/462111/
[12] *Stampede*. Creado por Benjamin Harvey Design. Disponible en https://freesound.org/people/benjaminharveydesign/sounds/350425/
[13] Yago Cordero Domenech: https://es.linkedin.com/in/yago-cordero-domenech-05ba6b88
[14] Se conoce como animación idle la que presenta el personaje cuando se encuentra inactivo, mostrando habitualmente un leve movimiento de brazos o simulando respirar, lo que lo dota de cierta vida y logra que no sea percibido como un elemento estático.

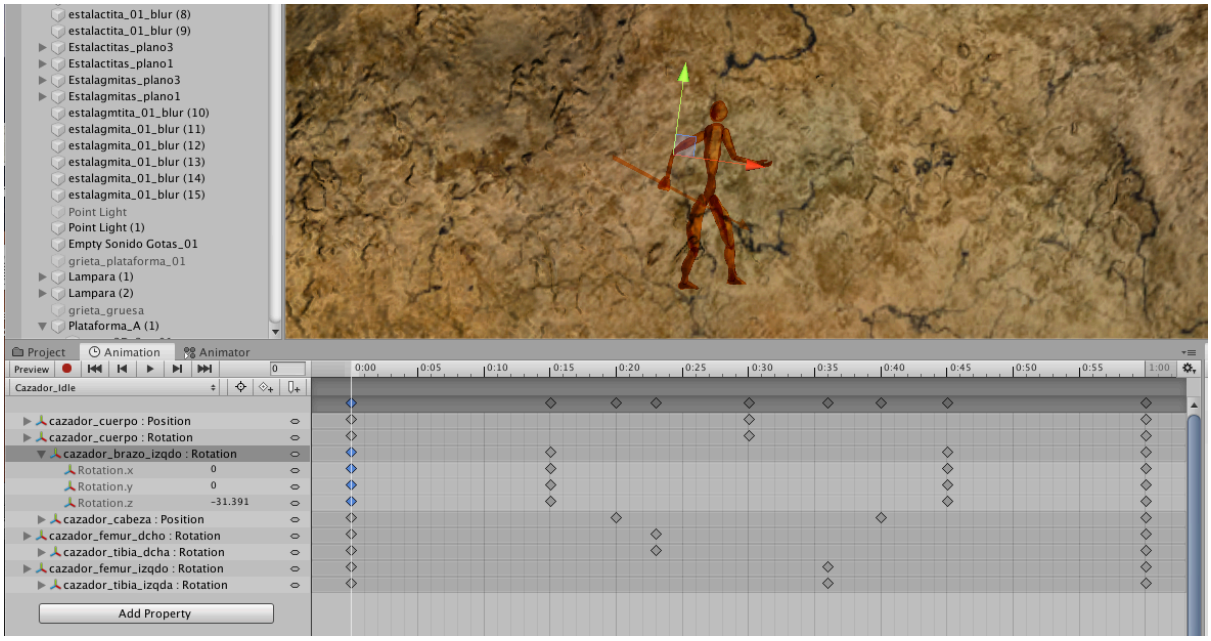


Fig. 6.6. Animando los ciclos del personaje

Una vez creados los cuatro ciclos que compondrán los diferentes comportamientos del personaje –idle, carrera, salto y aterriza– es necesario orquestar con el panel *Animator* su aparición según el personaje se encuentre desplazándose o estático, sobre una plataforma o en el aire, dando lugar al árbol lógico que podemos ver en la figura 6.7.

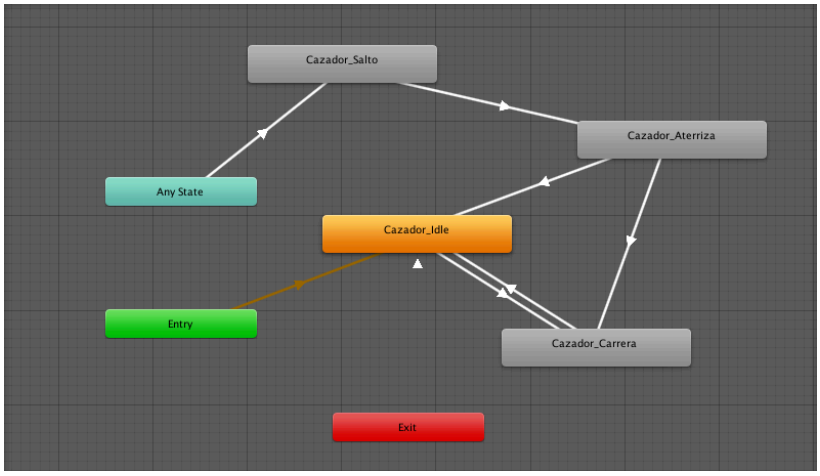


Fig. 6.7. Árbol lógico del comportamiento del personaje. Panel Animator

Además, el control del jugador debe ser programado mediante un script, que debe incluir las diversas condiciones que lanzarían los diferentes ciclos según el árbol lógico descrito (figura 6.8).

```
// Script Control Jugador
// Lenguaje: C#

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ControlJugador : MonoBehaviour{
```

```
    public float Velocidad = 2.5f;
    public float potenciaSalto = 7;
    private float entradaTeclado;
    private Rigidbody2D rb;
    private bool haciaDerecha = true;
    private bool enSuelo;
    public Transform enSueloCheck;
    public float radioCheck = 0.03f;
    public LayerMask esPlataforma;

private Animator anim;

    void Start(){
          anim = GetComponent<Animator>();
          rb = GetComponent<Rigidbody2D>();
    }

    void FixedUpdate(){

       enSuelo = Physics2D.OverlapCircle(enSueloCheck.position, radioCheck, esPlataforma);

           entradaTeclado = Input.GetAxis("Horizontal");

           rb.velocity = new Vector2(entradaTeclado*Velocidad, rb.velocity.y);

           if (entradaTeclado == 0) {
                    anim.SetBool("estaCorriendo", false);
           }
           else {
                    anim.SetBool("estaCorriendo", true);

           }

       if(haciaDerecha == false && entradaTeclado > 0){
          VoltearJugador();
       } else if(haciaDerecha == true && entradaTeclado < 0){
          VoltearJugador();
       }
    }

    void Update(){

       if (Input.GetKeyDown(KeyCode.Space) && enSuelo == true)
       {
          anim.SetTrigger("Saltar");
          rb.velocity = Vector2.up * potenciaSalto;
       }

       if (enSuelo == true)
       {
          anim.SetBool("estaSaltando", false);
       }
       else {
          anim.SetBool("estaSaltando", true);
       }
    }

    void VoltearJugador(){

       haciaDerecha = !haciaDerecha;
       Vector3 Escalar = transform.localScale;
       Escalar.x *= -1;
       transform.localScale = Escalar;
    }

}
```

Fig. 6.8. Script para el control completo del jugador

Los animales pintados en la superficie de la cueva –caballos, ciervas y bisonte– constituyen los únicos elementos vivos con los que el jugador va a interactuar, operando como plataformas móviles que le permiten acceder a espacios de otro modo inaccesibles. La animación de estos animales se importó fotograma a fotograma desde Krita, siguiendo las hojas de animación diseñadas por Carmen Pérez y María de Iracheta, e incorporadas en Unity a través del panel *Animation* (figura 6.9).

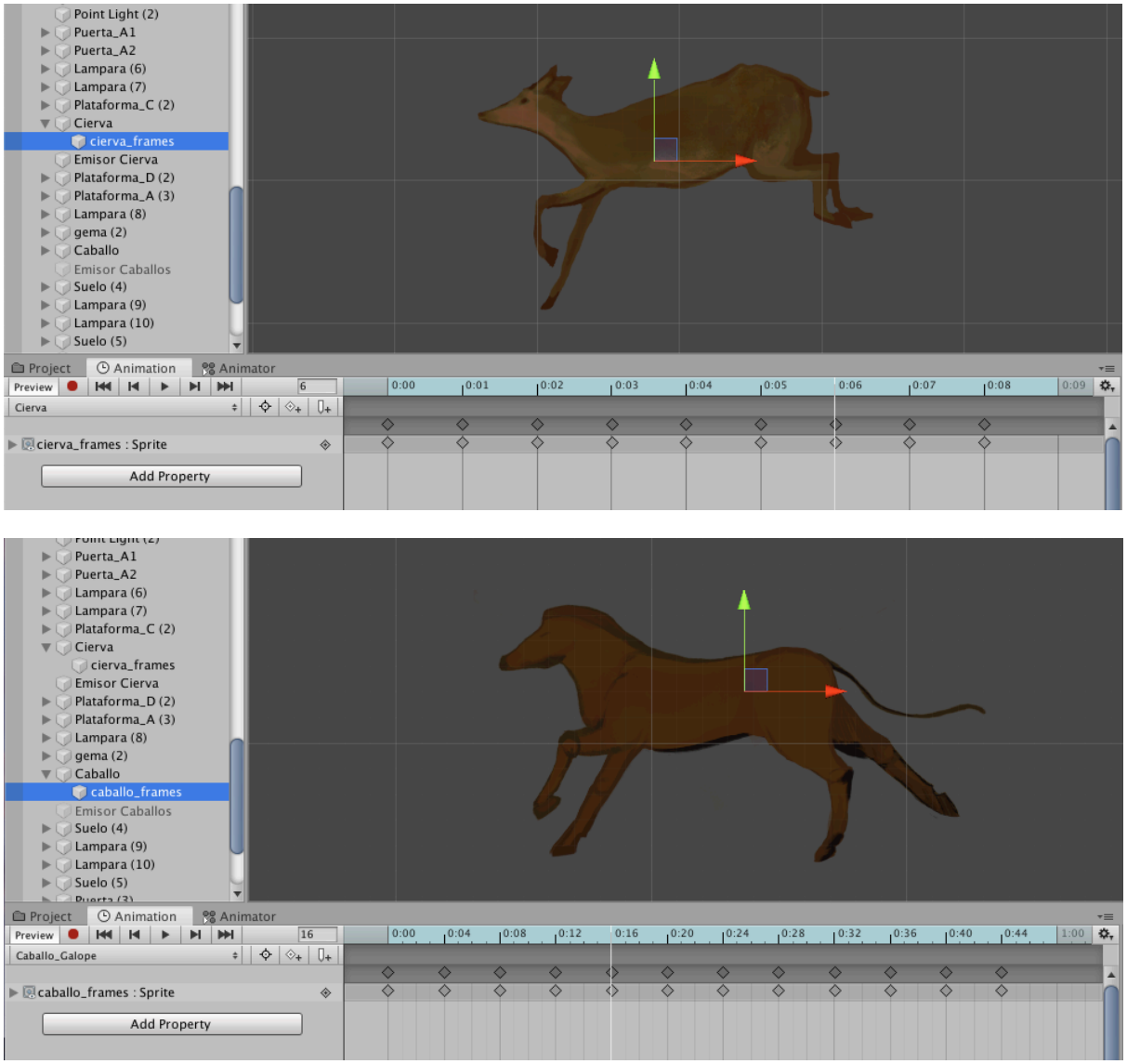


Fig. 6.9. Animación fotograma a fotograma de la cierva (superior) y el caballo (inferior)

Para que ambos animales operen como plataformas fue necesario agregarles a sus sprites un colisionador y un Platform Effector tal y como describimos para las plataformas de roca. De cara a que la imagen del animal se integrase con la pared y pareciera estar dibujada sobre esta, estas imágenes se aplicaron en modo de fusión Aditivo, funcionalidad no disponible en Unity y para cuya implementación fue necesario utilizar el shader *BlendModesFX* creado por Code Corsair[15].

El bisonte, enemigo final del nivel y guardián de la última puerta, fue también animado fotograma a fotograma desde Krita (figura 6.10) y además de operar como plataforma que permite al jugador saltar para acceder a elementos situados en altura, debe ser capaz de matar al jugador cuando le impacta con los cuernos. Para recrear este efecto situamos un colisionador en las astas del bisonte al que agregamos un script que destruye al jugador y recomienza el nivel desde el principio. También le añadimos un sonido de estampida que se intensifica al acercarnos al bisonte y un efecto de sacudida en la cámara, que introduce pequeños desplazamientos aleatorios en las coordenadas de esta en función de la distancia entre el cazador y el bisonte, simulando el temblor del suelo durante la estampida.

[15] Shader *BlendModesFX*. Creado por Code Corsair. Disponible en http://www.elopezr.com/photoshop-blend-modes-in-unity/

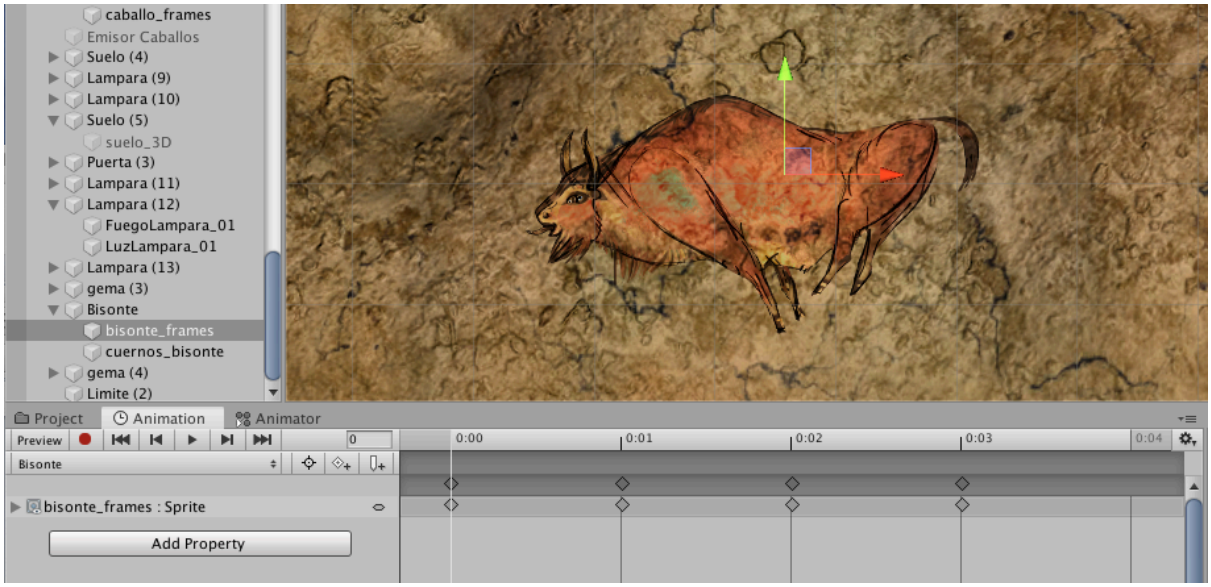


Fig. 6.10. Animación fotograma a fotograma del bisonte

Como elementos accesorios se añadieron una puerta de teletransporte que mediante un script traslada al personaje desde una ubicación en el nivel a otra, plataformas móviles que aparecen cuando el personaje coge algún ítem mágico, y un conjunto de estalactitas y estalagmitas, para las que diseñamos un script con el creamos un efecto de paralaje que las desplaza lateralmente a diferentes velocidades, incrementando con ello la sensación de profundidad.

En este punto sólo resta configurar el correcto comportamiento de la cámara durante el videojuego, elemento al que dedicaremos la siguiente Unidad Didáctica.

7. CONTROL Y GESTIÓN DE CÁMARAS EN JUEGOS 2D CON CINEMACHINE EN LA CREACIÓN DEL VIDEOJUEGO ALTAMIRA GAME

Lara Sánchez Coterón
Identificador ORCID: 0000-0002-3515-0979

Introducción

La cámara es uno de los elementos más importantes en el diseño de un juego, en casi cualquiera de los géneros, pero más aún si cabe en los videojuegos de plataformas 2D. Una cámara bien planteada permite al jugador focalizar en lo que está sucediendo y mostrarle las partes importantes del diseño. Por el contrario una cámara torpe puede arruinar toda la experiencia de juego. Es un objeto que afecta de manera estilística y sobre todo de modo funcional a la experiencia del jugador. La cámara puede ser utilizada con el objetivo de llamar la atención del jugador, proporcionándole suficiente información sobre el diseño y mostrándole lo que necesita ver para continuar la experiencia. Puede ofrecer un tipo de control directo del jugador sobre lo que se muestra en la pantalla, de tal manera que es el propio jugador quien decide qué es lo que quiere ver. Al mismo tiempo la cámara va facilitando y contextualizando los diferentes cambios que ocurren durante la experiencia de juego.
Concebir, pero sobre todo implementar la cámara es una de las partes más tediosas y complejas de este tipo de juegos. El enfoque más simple es fijar la cámara al personaje de juego para que esté siempre centrado en la pantalla. Esto es un fallo común entre principiantes con muchos inconvenientes a la hora de ofrecer determinada calidad durante el gameplay. Un efecto nocivo habitual en este uso de la cámara es el mareo innecesario que produce en los saltos. Si la cámara está fijada al jugador ésta se desplaza hacia arriba con cada salto del jugador, provocando dificultad en la lectura del entorno. Genera un efecto similar si el jugador comienza a correr en una dirección y se detiene o cambia de dirección repentinamente.

Altamira Game

En la creación del videojuego Altamira Game la evolución de la cámara ha ido sofisticando y mejorando la experiencia de juego. En las primeras versiones del juego se utilizó un primer script en lenguaje de programación C# llamado *Camera2DFollowShake.cs*.

Fig. 7.1. Captura de pantalla de las variables públicas del componente script *Camera 2D Follow Shake*

```
using System;
using UnityEngine;

namespace UnityStandardAssets._2D
{
   public class Camera2DFollowShake : MonoBehaviour
   {
      //variables públicas del camfollow
      public Transform target;
      public float damping = 1;
      public float lookAheadFactor = 3;
```

```
        public float lookAheadReturnSpeed = 0.5f;
        public float lookAheadMoveThreshold = 0.1f;

        //variables públicas del camshake
        public float Magnitud = 2;
        public GameObject Objeto1;
        public GameObject Objeto2;
        public float DistanciaAtenuacion = 2;
        public GameObject Enemigo;

        private float m_OffsetZ;
        private Vector3 m_LastTargetPosition;
        private Vector3 m_CurrentVelocity;
        private Vector3 m_LookAheadPos;
        float deltaX;
        float deltaY;
        float shake;
        //private GameObject bisonte;

        // Use this for initialization
        private void Start()
        {
            m_LastTargetPosition = target.position;
            m_OffsetZ = (transform.position - target.position).z;
            transform.parent = null;

            //bisonte = GameObject.Find("Bisonte");
        }

        // Update is called once per frame
        private void Update()
        {

            // only update lookahead pos if accelerating or changed direction
            float xMoveDelta = (target.position - m_LastTargetPosition).x;

            bool updateLookAheadTarget = Mathf.Abs(xMoveDelta) > lookAheadMoveThreshold;

            if (updateLookAheadTarget)
            {
                m_LookAheadPos = lookAheadFactor*Vector3.right*Mathf.Sign(xMoveDelta);
            }
            else
            {
                m_LookAheadPos = Vector3.MoveTowards(m_LookAheadPos, Vector3.zero, Time.deltaTime*l
ookAheadReturnSpeed);
            }

            Vector3 aheadTargetPos = target.position + m_LookAheadPos + Vector3.forward*m_OffsetZ;
            Vec-
tor3 newPos = Vector3.SmoothDamp(transform.position, aheadTargetPos, ref m_CurrentVelocity, dampi
ng);

            float distanceBetweenObjects = Vector2.Distance(Objeto1.transform.position, Objeto2.transfor
m.position);

            if (distanceBetweenObjects <= DistanciaAtenuacion && Enemigo.activeSelf == true) {
                shake = Magnitud/distanceBetweenObjects;
            }
            else {
                shake = 0;
            }

            deltaX = shake*(UnityEngine.Random.Range(-1f, 1f));
            deltaY = shake*(UnityEngine.Random.Range(-1.5f, 1.5f));

            transform.position = newPos + new Vector3 (deltaX, deltaY, 0);

            m_LastTargetPosition = target.position;

        }
    }
}
```

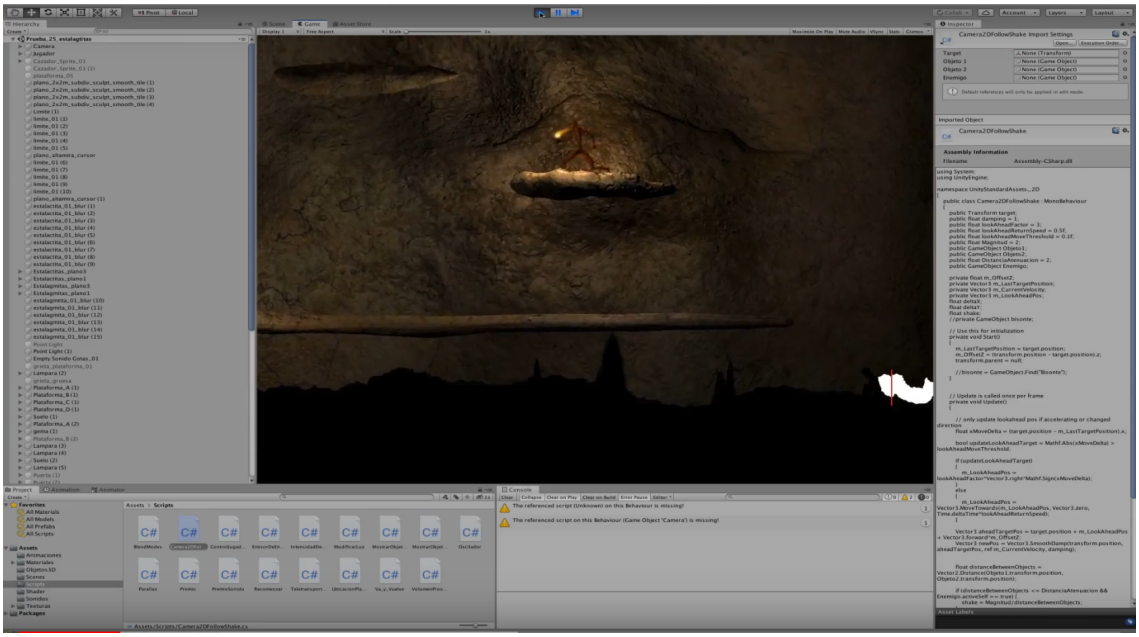


Fig. 7.2. Captura de pantalla del prototipo alpha del videojuego Altamira Game v1.0. desarrollado en la versión gratuita del motor de videojuegos Unity, distribución 2019.2.6f1

Sin embargo este control de cámara acarreaba una serie de debilidades que rápidamente fueron identificadas en sesiones de prueba del prototipo alpha Altamira_v1.0. Con un valor de Damping=5 simétrico en ambos ejes la amortiguación de la cámara hacía un seguimiento extremadamente lento del jugador. Esto hacía que el personaje de juego se saliese de pantalla con facilidad, produciendo dificultad al acceso de informaciones básicas durante el gameplay (emplazamiento de las plataformas, de los ítems, de personajes no jugables como el enemigo…). Este tipo de control hacía que la cámara tardase mucho en reaccionar a los cambios de dirección. Este hecho además conllevaba situaciones altamente frustrantes para el jugador como el ataque del enemigo final del nivel mucho antes de que el jugador lo hubiese podido visualizar debido a ese retraso de la cámara sobre el personaje jugable.

El script de partida además no contemplaba la parametrización diferencial entre ejes (X, Y). El ritmo en el diseño de niveles en horizontal y en vertical era diferente, mucho más lento en eje Y, donde el reto de desplazarse en vertical atenuaba ligeramente esa sensación de lentitud de la cámara. En ambos casos seguíamos percibiendo momentos de error en los que el jugador perdía de vista a su personaje jugable, tanto subiendo y descendiendo plataformas, como desplazándose hacia izquierda y derecha. Este error es fácilmente corregible usando un valor de amortiguación inferior, cercano al 0.5, sin embargo la asimetría de ritmo entre el diseño de niveles horizontal y vertical hace necesario poder diferenciar estos valores de seguimiento amortiguado de la cámara en ambos ejes por separado.

En algunas ocasiones en un equipo formado mayormente por artistas y diseñadores lidiar con las lógicas y la sintaxis de un lenguaje de programación como C# es algo más lento y farragoso que usar objetos predefinidos que contienen características modulables. En el diseño de Altamira Game hemos optado por utilizar un desarrollo que desde noviembre de 2017 se puede sumar a los proyectos realizados en Unity, para versiones 2017.1 y posteriores, que solventa de manera rápida y eficaz la gestión y el control de las cámaras de proyectos 2D y 3D. Estamos hablando de Cinemachine.

Cinemachine es un paquete que contiene un conjunto de módulos para gestionar las cámaras en el motor de videojuegos Unity, resolviendo lógicas complejas de seguimiento de objetivos, composición y otros parámetros de cámara sin tener necesidad de manipular scripts en C#. La naturaleza procedural de sus módulos hace que Cinemachine pueda ajustar dinámicamente el comportamiento de las cámaras para perseguir el tipo de toma determinada por el usuario. Tiene alta compatibilidad con otras herramientas de Unity y ofrece la posibilidad de crear extensiones propias o integrarlas con scripts de cámara personalizados en C#.

Instalación de Cinemachine en Unity

Una vez creado o abierto el proyecto en el que deseamos trabajar, descargaremos el paquete desde el menú de la parte superior izquierda del motor: Window/Package Manager. Se nos abrirá una ventana emergente como la que vemos en la imagen inferior (figura 7.3). En la parte izquierda de esa ventana se cargan todos los paquetes disponibles para el motor, listados en orden alfabético. Seleccionamos Cinemachine y presionamos el botón

"Install" que aparece en la parte inferior derecha de la ventana emergente. Una vez instalado el paquete en esa misma área nos aparecerán dos botones "Up to date", activo en el caso de que haya actualizaciones del paquete y "Remove" para eliminar el paquete del proyecto.

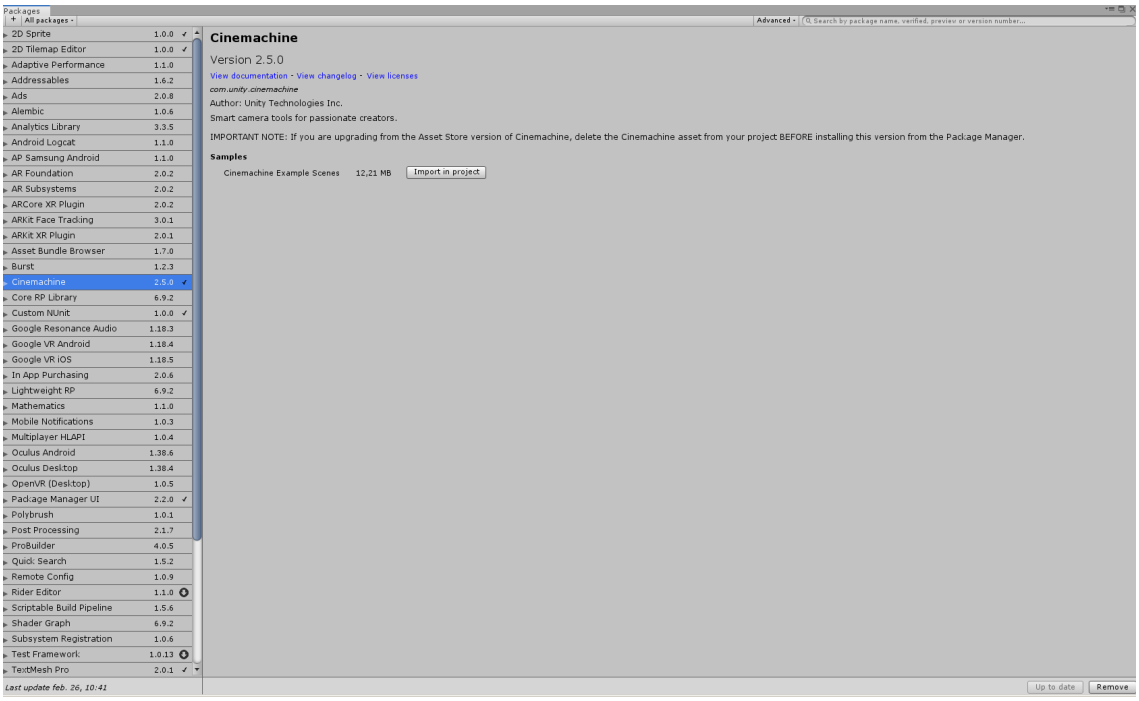


Fig. 7.3. Captura de pantalla de la ventana emergente del Package Manager de Unity. La versión disponible en febrero 2020 es Cinemachine 2.5.0

Una vez instalado Cinemachine nos aparecerá una nueva opción de menú en la barra superior del motor, entre el menú Component y el menú Window> Cinemachine (figura 7.4).

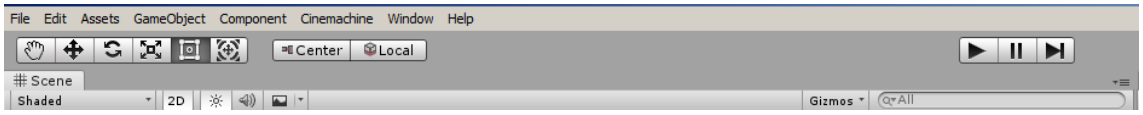


Fig. 7.4. Captura de pantalla del menú principal del motor Unity en un proyecto con Cinemachine instalado

Configuración de Cinemachine para la cámara 2D en el juego de plataformas Altamira Game

El estado en el que se encuentra el proyecto Altamira Game, en su versión 1.0,el juego es funcional, es decir, podemos desplazarnos por las plataformas y por el mapa con las teclas de flechas izquierda y derecha, saltar con la tecla espacio y la cámara sigue al jugador tal y como hemos descrito anteriormente. La "Main Camera" ha sido renombrada como "Camera". Accedemos al menú propio del paquete Cinemachine para crear una nueva cámara virtual 2D en la siguiente ruta; CINEMACHINE/Create 2D camera. Aparece un nuevo objeto en el panel Hierarchy llamado "CM vcam1", es decir una cámara virtual de cinemachine.

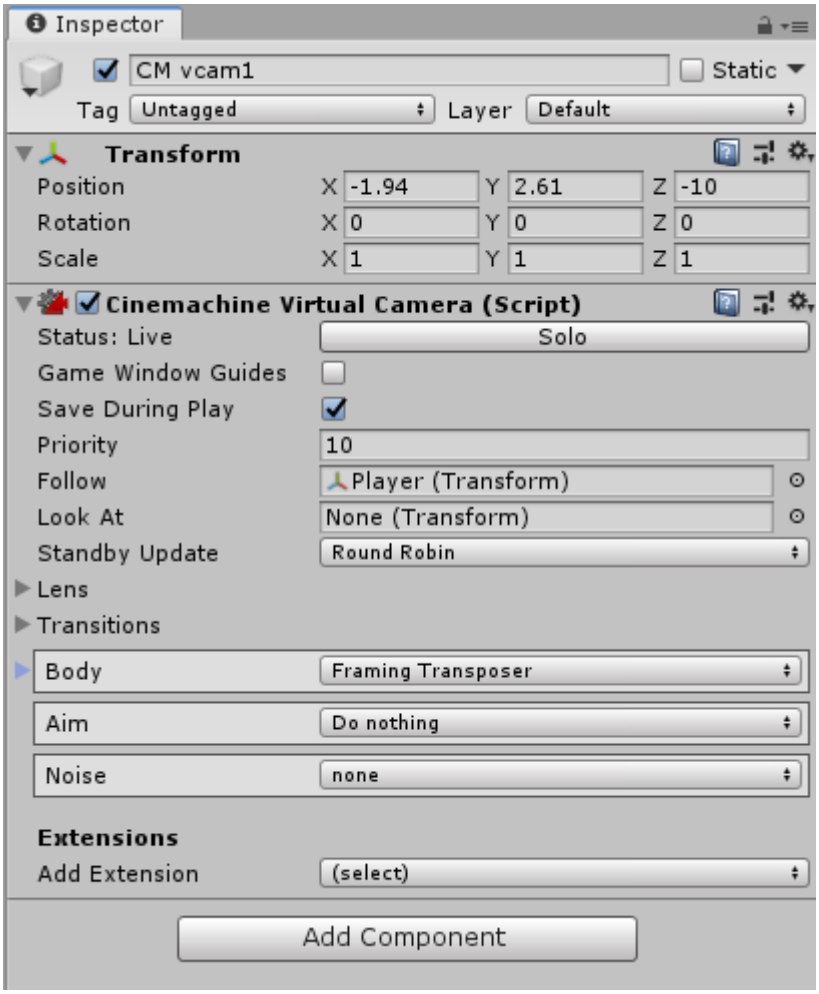


Fig. 7.5. Captura de pantalla de los componentes y propiedades de la Cámara Virtual 2D de Cinemachine, en la vista del panel Inspector

Vamos a renombrar este objeto como “CM vcam regular” para diferenciarlo más delante de otras cámaras virtuales que crearemos dentro de la misma escena del proyecto. Uno de los usos de las cámaras virtuales es que podemos ir vinculando nuestra mainCamera a diferentes cámaras virtuales en escena. El objeto “CM vcam regular” está compuesto por un componente TRANSFORM (común a todos los objetos en Unity) y un componente de parametrización llamado CINEMACHINE VIRTUAL CAMERA (SCRIPT) (figura 7.5)

Estos valores se asignan por defecto a nuestra “Main Camera”, de tal manera que al crear la cámara virtual se añade un nuevo componente a nuestra “Main Camera” denominado Cinemachine Brain (figura 7.6)

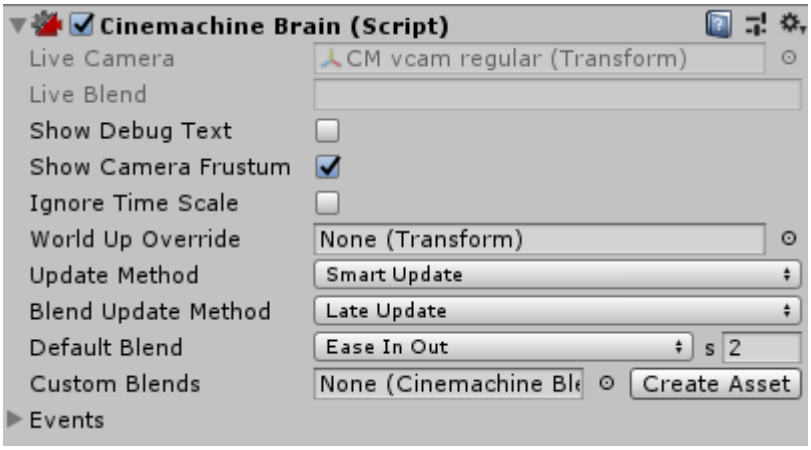


Fig. 7.6. Captura de pantalla del componente Cinemachine Brain que se añade a la cámara principal del proyecto (Vista parcial del panel Inspector)

Entre las variables públicas del componente Cinemachine Virtual Camera (Script) (figura 7.5) tenemos la opción de que la cámara siga cualquier objeto en escena. En este caso la cámara va a seguir al objeto Player. Para ello arrastramos el objeto desde el panel Hierarchy hasta en slot "Follow".
Podemos utilizar diferentes propiedades del apartado Body (figura 7.5) para especificar el algoritmo que mueve la cámara virtual en la escena. Cinemachine incluye seis algoritmos distintos para gestionar una cámara virtual:

- *Transposer:* se mueve en una relación fija con el objetivo a seguir.
- *Do Nothing:* no mueve la cámara virtual.
- *Framing Transposer:* se mueve en una relación fija entre espacio y pantalla con el objetivo a seguir.
- *Orbital Transposer:* se mueve en una relación variable con el objetivo a seguir, con la posibilidad de aceptar opciones de control del jugador.
- *Tracked Dolly:* se mueve a lo largo de un recorrido predefinido.
- *Hard Lock to Target:* utiliza la misma posición en el objetivo a seguir.

En nuestro proyecto hemos elegido utilizar el algoritmo Framing Transposer dado que ofrece una serie de funciones interesantes para el diseño de plataformas 2D que tenemos entre manos. Este algoritmo mueve la cámara en una relación de espacio de pantalla fija al objetivo a seguir. También puede especificar compensaciones, amortiguación de movimiento y reglas de composición. Este algoritmo solo varía la posición de la cámara en el espacio, no reorienta ni apunta la cámara en otras direcciones. Esta es una opción diseñada mayormente para cámaras 2D y ortográficas, aunque funciona igualmente bien con cámaras en perspectiva y entornos 3D. Es importante tener en cuenta que para usar Framing Transporter, la propiedad "Look At" debe estar vacía.
Vamos a reparar en cuatro de los bloques de parámetros que nos ofrece este algoritmo de control: Lookahead, Damping, Dead Zone y Soft Zone (figura 7.7).

Lookahead time: Ajusta el desplazamiento de la cámara virtual, respecto del personaje de juego, en relación al movimiento del mismo. Cinemachine estima el punto en el que el jugador estará tantos segundos en el futuro (desde cero hasta 1 seg). Sirve para que el jugador tenga visibilidad anticipada en el eje en el que se mueve.
Damping: La capacidad de respuesta de la cámara para mantener el desplazamiento en el eje correspondiente (x, y, z). Con valores cercanos al cero la cámara será más receptiva a los desplazamientos en estos ejes. Con valores cercanos al 20 la cámara responderá más lentamente a la amortiguación.

Ambas variables están presentes en el script inicial del prototipo, pero la capacidad de parametrización de las mismas es mucho menor (figura 1). Además tenemos la posibilidad de gestionar otros comportamientos interesantes en la misma cámara como zonas muertas (Dead Zone) y de acomodamiento (Soft Zone).

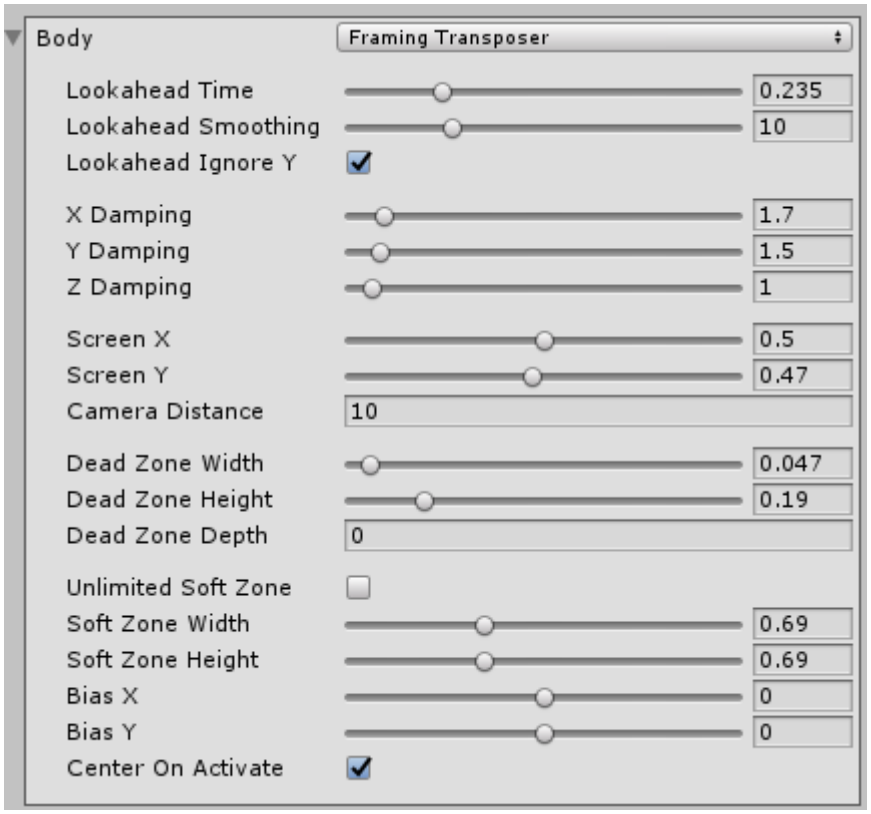


Fig. 7.7. Captura de pantalla del apartado Body dentro del componente CINEMACHINE VIRTUAL CAMERA (SCRIPT) de la cámara virtual

Dead Zone: delimita un área (x, y) en la que el personaje de juego se puede mover sin que se mueva la cámara. Esto genera una cierta estabilidad en zonas en las que el cometido del jugador puede estar más ligado a la exploración que de consumo del espacio y genera diferentes ritmos de avance en el diseño general del nivel.
Soft Zone: Si el personaje de juego entra en esta región del encuadre, la cámara se reorientará para volver a colocarlo en la Dead Zone. Lo hará lenta o rápidamente, de acuerdo con el tiempo especificado en la configuración de amortiguación (Damping).

Esta cámara virtual de Cinemachine no solo amplía las posibilidades de gestión y control de nuestra "Main Camera" sino que nos ofrece la posibilidad de replicar la segunda parte del script de partida, es decir, la función del CamShake. La cámara inestable, cámara en mano o cámara libre es una técnica que proviene del ámbito cinematográfico en la cual los métodos convencionales de estabilización de la imagen están decididamente suprimidos en aras de premisas semánticas concretas. Se trata de un recurso que se utiliza de modo expresivo también en el ámbito de los videojuegos de diferentes modos. En el diseño del prototipo de Altamira Game se usa para reforzar la presencia del personaje no jugable final: el gran bisonte.

Uso de Noise Properties para introducir el cam shake en Altamira Game.

```
//variables públicas del camshake
      public float Magnitud = 2;
      public GameObject Objeto1;
      public GameObject Objeto2;
      public float DistanciaAtenuacion = 2;
      public GameObject Enemigo;
```

Para sustituir las funciones de camshake del componente script "Camera 2D Follow Shake" utilizado en el primer prototipo del juego, hemos usado las propiedades Noise de las cámaras virtuales de Cinemachine. En concreto el perfil "6D Shake" con los valores por defecto del componente.

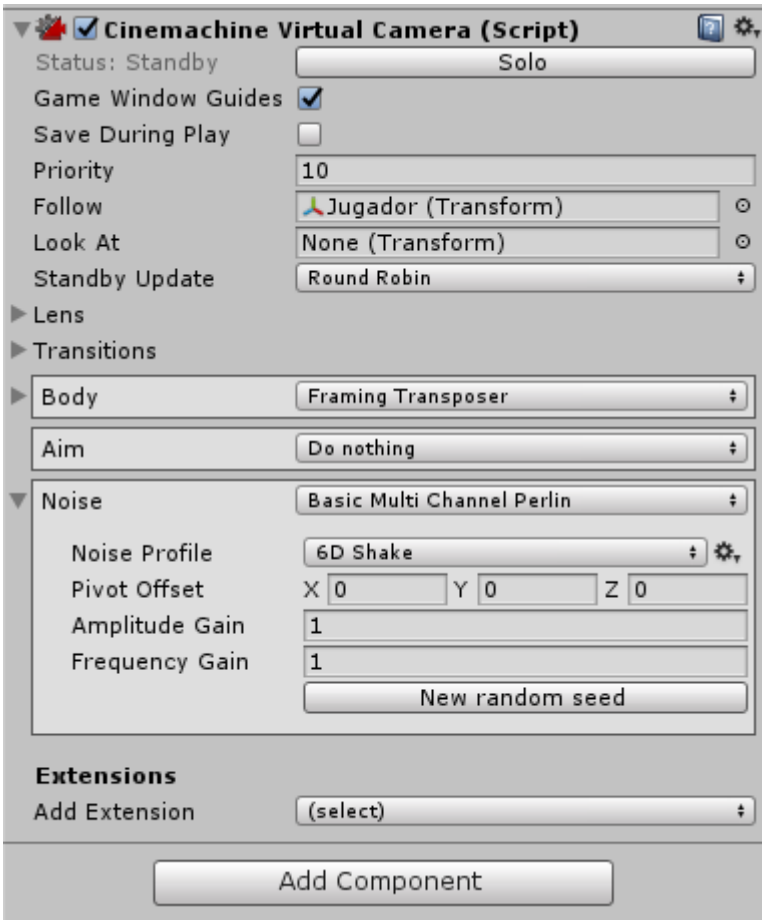


Fig. 7.8. Captura de pantalla en la que podemos apreciar el apartado Noise desplegado, dentro del componente CINEMACHINE VIRTUAL CAMERA (SCRIPT) de la cámara virtual

Vinculación y transición de diferentes cámaras virtuales a la misma mainCamera vinculado a eventos animados: uso de las State-Driven Cameras

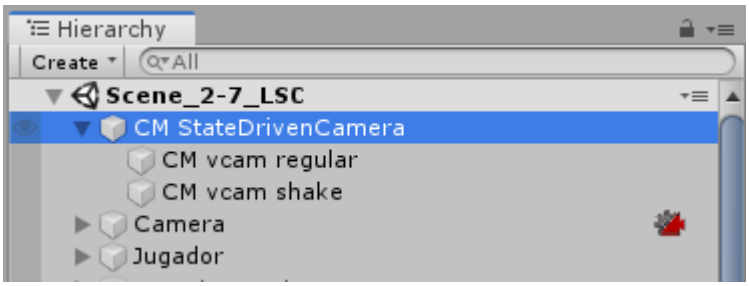


Fig. 7.9. Captura de pantalla en la que podemos ver el objeto CM StateDrivenCamera con dos cámaras virtuales anidadas. Ambas han sido renombradas, una como CM vcam regular y otra como CM vcam shake

El objeto Cinemachine State-Driven Camera sirve para activar cámaras virtuales secundarias cuando un objeto con animación cambia de estado. En entornos 3D se puede utilizar, entre otras situaciones, para que la cámara del personaje de juego tenga implementado un cam-shake mientras corre y otra cámara más amortiguada mientras camina. Para crear una State-Driven Camera, desde el menú principal del motor, elegimos Cinemachine/Create State-Driven Camera. Esto genera el objeto CM StateDrivenCamera en el panel Hierarchy con una nueva cámara virtual anidada (figura 7.9).
Desde el panel Inspector, en el componente Cinemachine State Driven Camera (Script) (figura 7.10), asignamos en la variable pública Animated Target el objeto de animación que va a funcionar como desencadenante, en este

caso el objeto “bisonte_frames (Animator)”. Esto va a hacer que cada vez que el Animator Controller del objeto “bisonte_frames”, nombrado como “bisonte_01” en el proyecto, este activo, el objeto CM State-Driven Camera va a poner en funcionamiento la cámara virtual CM vcam shake (figura 7.10). En nuestro caso el cam-shake está vinculado al momento final del nivel, en el área del enemigo final con lo cual no tenemos necesidad de cambiar otra vez a una cámara virtual anterior, pero en el caso de necesitarlo podemos asignarlo desde el apartado State del componente Cinemachine State Driven Camera (Script).

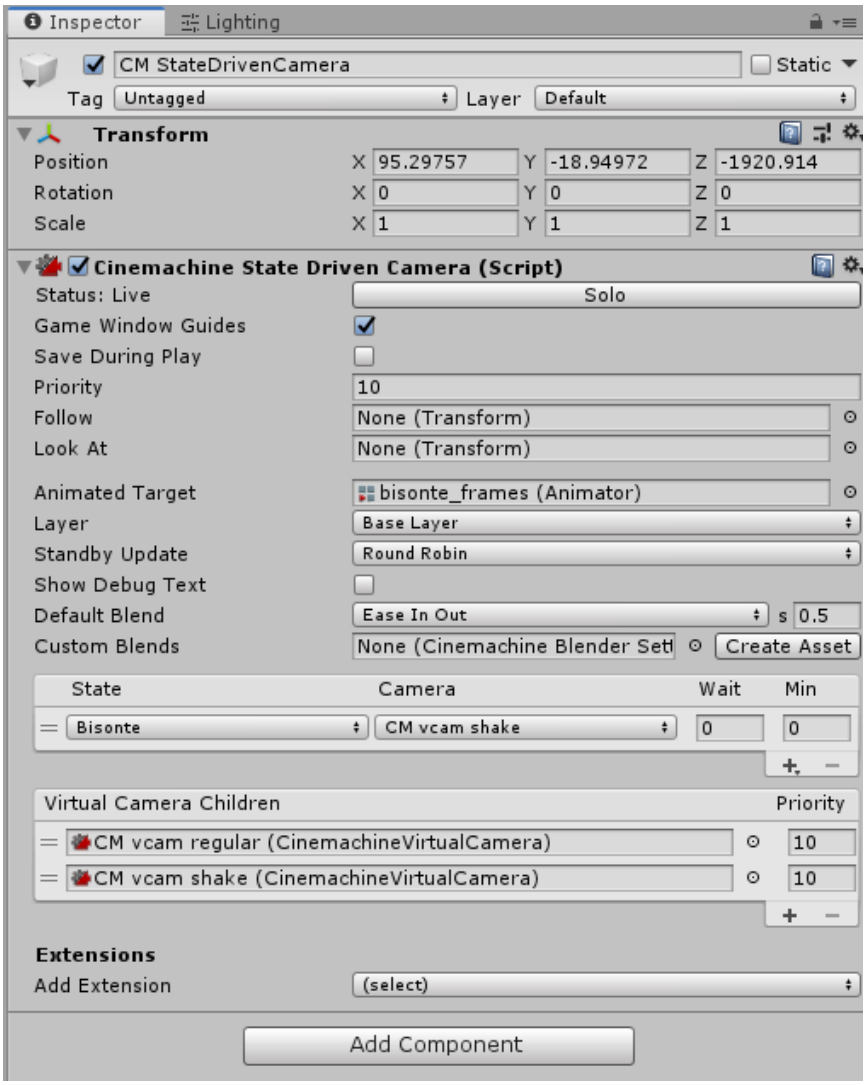


Fig. 7.10. Captura de pantalla del panel inspector del objeto CM StateDrivenCamera

Referencias

- Itay Keren (2015) *Scroll Back: The Theory and Practice of Cameras in Side-Scrollers*. Game Developers Conference. San Francisco, CA. consultado el 19 feb 2020 https://gdcvault.com/play/1022243/Scroll-Back-The-Theory-and

- Documentación general oficial del motor:
https://docs.unity3d.com/Packages/com.unity.cinemachine@2.2/manual/index.html

- Framing Transposer:
https://docs.unity3d.com/Packages/com.unity.cinemachine@2.2/manual/CinemachineBodyFramingTransposer.html

- Noise Properties:
https://docs.unity3d.com/Packages/com.unity.cinemachine@2.5/manual/CinemachineVirtualCameraNoise.html

- State Driven Cameras:
https://docs.unity3d.com/Packages/com.unity.cinemachine@2.5/manual/CinemachineStateDrivenCamera.html

- Using Cinemachine: State Driven Cameras https://www.youtube.com/watch?v=2X00qXErxIM

8. LA DOCUMENTACIÓN Y SU DIFUSIÓN EN EL PROCESO DE DISEÑO Y CREACIÓN DE VIDEOJUEGOS

Marcos Casero Martín
Identificador ORCID: 0000-0001-6937-8309

Introducción

La fase de documentación para el diseño, creación y ambientación de un videojuego es un proceso importante a tener en cuenta si se quiere dotar al mismo de elementos que aporten al jugador información veraz y, en el caso del videojuego educativo, pedagógica.

En el caso del presente proyecto llevado a cabo bajo el título "Diseño y creación de videojuegos mediante herramientas de software libre" el equipo se ha centrado en el Arte Rupestre como eje temático y más en concreto en las Cuevas de Altamira. No sólo se ha desarrollado siguiendo una estética que se asemeja a las pinturas paleolíticas que podemos encontrar en el interior de la cueva, sino que además a lo largo del juego aparecen pop-ups explicativos acercando al usuario a la vida prehistórica mediante datos relativos a las costumbres, modos de vida y herramientas de la época.

Para la documentación y recolección de información se ha recurrido a diferentes fuentes. Por un lado, se encuentran los textos escritos. Entre ellos se han consultado algunos específicos y centrados en las Cuevas de Altamira como *Altamira y otras cuevas de Cantabria* de Miguel Ángel García Guinea y otros que hacen referencia a otras cuevas prehistóricas como *Conoce Tito Bustillo* de Alfonso Millara. Por otro lado, se ha contactado con expertos en la materia como el catedrático Pedro Saura Ramos especializado en arte rupestre paleolítico y en las Cuevas de Altamira.

Para la difusión de la información obtenida sobre el mundo del arte rupestre, se han seguido dos líneas de actuación: una enfocada en la experiencia de juego de la que disfrutará el usuario mediante mensajes con explicaciones y curiosidades y otra dirigida principalmente a la comunidad universitaria a través del formato de conferencia.

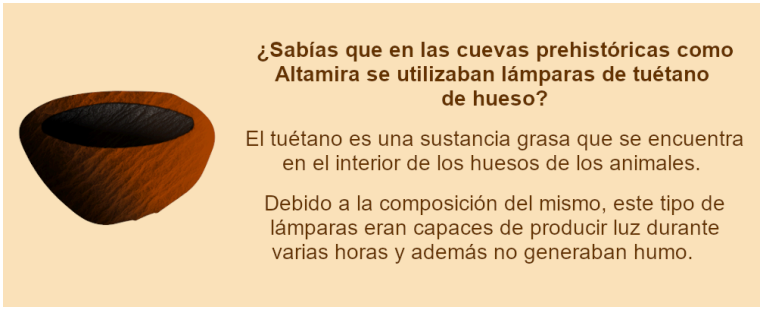
Asimismo, entre los objetivos del proyecto, se encuentra acercar al alumno al mundo de la industria del videojuego, por lo que también se ha programado otra conferencia en torno a este tema.

Mensajes pop-up explicativos

Como se ha indicado anteriormente, se pretende que la experiencia de juego no solo sea de entretenimiento, sino que sea enriquecedora a nivel educativo. Desde un primer momento el grupo del trabajo planteó que era necesario incluir un enfoque pedagógico al proyecto. Por este motivo se decidió que era importante la introducción en el videojuego de mensajes pop-up de carácter explicativo que aportasen datos veraces sobre cómo era la vida prehistórica. De este modo, por ejemplo, el jugador podría conocer información relativa a las herramientas que se utilizaban en la época o al descubrimiento del fuego.

Siguiendo la metodología y los objetivos planteados, estos mensajes se han diseñado gracias a herramientas de software libre; en este caso el editor de imágenes GIMP.

Para el nivel de juego diseñado, se han creado tres mensajes pop-ups. Uno que da la bienvenida al juego y dos explicativos en relación a objetos que el usuario puede encontrar a lo largo de su recorrido. El primero de ellos hace referencia a las lámparas de tuétano de hueso, utilizadas por el hombre prehistórico para tener iluminación dentro de las cuevas, puesto que no producían humo y su luz era capaz de aguantar durante largos periodos de tiempo. El segundo trata sobre el sílex, uno de los materiales más utilizados de la época para la creación de herramientas y utensilios (figura 8.1).



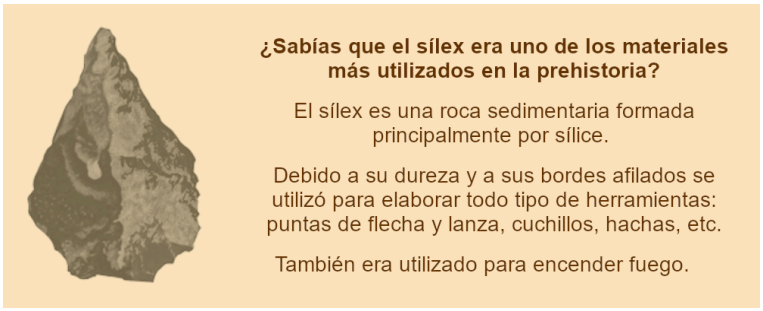


Fig. 8.1. Cuadro explicativo sobre las lámparas de tuétano (superior) y las herramientas de sílex (inferior)

Para el diseño de estos cuadros se han tomado los props elaborados previamente por el equipo y se han añadido posteriormente los textos explicativos. Para ello, mediante el programa de software libre GIMP, se ha creado un archivo de 1000 píxeles de ancho por 400 de alto. Para aplicar el color al fondo de la imagen se ha utilizado la *Herramienta de relleno*, el objeto se ha copiado de la imagen original y se ha pegado en una capa nueva para luego escalarla y ponerla en el lugar apropiado mediante la *Herramienta de transformación unificada*. Por último, el texto se ha introducido con la *Herramienta de texto*. El archivo se ha guardado en formato PNG para ser insertado en el nivel de juego creado.

Conferencia sobre arte paleolítico

La difusión de la información relativa al arte rupestre se ha complementado con la conferencia impartida por el catedrático Pedro Saura Ramos bajo el nombre *El arte paleolítico y la recreación de Altamira* (figura 8.2).

Fig. 8.2. Cartel de la conferencia *El arte paleolítico y la recreación de Altamira*

Esta conferencia tuvo lugar el jueves 23 de enero de 2020 en el salón de grados de la Facultad de Bellas Artes de la Universidad Complutense de Madrid y era de acceso libre tanto para la comunidad universitaria como para toda persona ajena a la misma que tuviera interés por esta. Fue difundida por los canales oficiales de la Facultad y la convocatoria fue un éxito llenando el espacio destinado a dicha actividad (figura 8.3).

Fig. 8.3. Pedro Saura durante la conferencia *El arte paleolítico y la recreación de Altamira*, celebrada el 23 de enero de 2020

Pedro Saura Ramos es un experto en arte paleolítico, en su documentación fotográfica y en la reproducción de facsímiles de arte rupestre. Tal y como recoge su currículum es el director y autor junto con Matilde Múzquiz, de la réplica del techo policromo de la cueva de Altamira, instalado en el nuevo Museo de Altamira. Además de este también ha dirigido y creado otros facsímiles de diversos yacimientos del norte de España. Actualmente es profesor emérito en el Departamento de Diseño e Imagen de la Facultad de Bellas Artes de la Universidad Complutense de Madrid.
Durante su conferencia acercó el mundo del arte rupestre a los asistentes repasando cómo era la vida del hombre de la prehistoria contando anécdotas y curiosidades al público. Hizo un recorrido por algunas de las cuevas más importantes de España que presentan vestigios de arte paleolítico como la Cueva de la Garma, las Cueva de El Castillo, la Cueva de El Mirón o la Cueva de El Pendo.
Por otro lado, habló del arte rupestre indicando este que no sólo se trataba de pinturas en el interior de las cuevas, ya que también podían encontrarse en el exterior y cerca de los ríos. Asimismo, apuntó que una parte importante del arte de la época se trataba de arte mueble, útiles tallados sobre huesos, astas, dientes o sílex. Especial interés tienen las cabezas de cierva talladas en diferentes materiales como huesos o cuernos.
Por último, se centró en la Cueva de Altamira: en su origen, su historia, sus pinturas y algunas curiosidades como la distancia real a la que se encontraba el suelo y el techo de la misma en un primer momento.
Tanto con esta conferencia como con los mensajes pop-ups explicativos, se pretende acercar el conocimiento del arte rupestre a toda persona interesada en el tema e incluir el enfoque pedagógico en el proyecto a desarrollar, haciendo hincapié en que el formato del videojuego no se trata únicamente de un medio de entretenimiento, sino que también es una herramienta educativa.

Conferencia sobre la industria del videojuego

Acercar al estudiante a la industria del videojuego dando a conocer sus metodologías de trabajo era uno de los objetivos de este proyecto. Por este motivo se programó una conferencia impartida por Miguel Pascual Romero que llevaba por título *Diseñando entornos en un videojuego triple A: Desde el concept art al arte final* (figura 8.4).

Fig. 8.4. Cartel de la conferencia *Diseñando entornos en un videojuego triple A: Desde el concept art al arte final*

Esta conferencia se desarrolló en el salón de grados de la Facultad de Bellas Artes de la Universidad Complutense de Madrid el 12 de diciembre de 2019 y era de acceso libre (figura 8.5.). Se difundió por los medios oficiales de la Facultad de Bellas Artes y entre los estudiantes de los profesores que forman el equipo de este proyecto y que imparten asignaturas de perfil artístico-tecnológico en los grados de Bellas Artes y Diseño de Videojuegos.

Fig. 8.5. Miguel Pascual durante la conferencia *Diseñando entornos en un videojuego triple A: Desde el concept art al arte final*

Miguel Pascual es experto en el mundo de los videojuegos. Actualmente, es Lead Environment Artist en la compañía Mercury Steam. Le avala una larga experiencia en el sector audiovisual: ha trabajado en la película de animación “Planet 51”, en la producción de títulos como la trilogía "Castlevania: Lords of Shadow", “Space-lords" o "Metroid: Samus Returns" y lleva más de 10 años en el desarrollo de videojuegos.
Durante la conferencia, acercó el mundo de esta industria a los asistentes comentando el día a día en un estudio de creación de videojuegos y las distintas fases de desarrollo de un proyecto de este tipo desde los primeros pasos hasta el resultado final. Asimismo, se expusieron cuáles eran algunas de las cualidades más valoradas en la industria a la hora de contratar a nuevos profesionales. Con todo ello se mostró una visión global de cómo se encuentra el sector en la actualidad.

9. TABLA RESUMEN DE HERRAMIENTAS LIBRES Y RECURSOS ONLINE PARA EL DISEÑO DE VIDEOJUEGOS

Para concluir el presente estudio incluimos una tabla resumen de las herramientas y recursos utilizados a lo largo de este proyecto y que pueden ser de utilidad en el diseño y creación de videojuegos, así como su consideración como software libre o gratuito.

Recurso	**Descripción**	**Software libre / Gratuito**
Gravit Designer	Programa online de diseño vectorial	Gratuito
Autotracer	Programa online de trazado vectorial	Gratuito
Krita	Dibujo y animación digital	Software libre
GIMP	Diseño digital	Software libre
Inkscape	Diseño vectorial	Software libre
Blender	Diseño 3D	Software libre
Audacity	Editor de audio	Software libre
Unity	Diseño de videojuegos	Gratuito
C#	Lenguaje de programación	Gratuito
Cinemachine	Gestor de cámaras para Unity	Gratuito
TextureHaven.com	Biblioteca de texturas	Software libre
SoundBible.com	Biblioteca de sonidos	Gratuito
FreeSound.org	Biblioteca de sonidos	Gratuito
BlenderModesFX	Shader de Unity con Modos de Fusión	Gratuito

10. CONCLUSIÓN DEL PROYECTO

Una vez finalizado el proyecto hemos podido corroborar la idoneidad de las herramientas de software libre y gratuitas para el diseño y desarrollo de videojuegos. Nuestra experiencia, en la que creamos un nivel jugable de un videojuego, puede fácilmente ampliarse hasta desarrollar un producto completo con tan sólo repetir el proceso tantas veces como niveles se desee que tenga el proyecto.

El videojuego desarrollado en este PIMCD, *Altamira Game*, puede jugarse online en la siguiente dirección web[16]:

https://www.ucm.es/altamiragame/

Importante: Para poder visualizar el videojuego adecuadamente es necesario utilizar navegador Google Chrome (versión 56 o superior) o Mozilla Firefox (versión 51 o superior), y tener activada compatibilidad con WebGL. Puedes seguir los pasos descritos en estos tutoriales para activar WebGL en Chrome[17] o en Firefox[18].

[16] De cara a ofrecer un respaldo online, también se ha dispuesto otra copia jugable del videojuego en la dirección web: http://www.dariolanza.com/altamiragame
[17] https://www.spoots.com/activar-webgl-en-chrome
[18] https://blog.trescomatres.com/2013/12/habilitar-webgl-en-navegador-firefox/


UNIVERSIDAD
COMPLUTENSE
MADRID